(In)security of efficient tree-based group key agreement using bilinear map

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Citation: PHAN, R.C.-W., and GOI, B.-M., 2008. (In)security of efficient tree-based group key agreement using bilinear map. IN: IEEE/IFIP International Embedded and Ubiquitous Computing, (EUC '08), Shanghai, 17-20 Dec., vol. 2, pp. 443 - 446

Additional Information:

- This is a conference paper [© IEEE]. It is also available at: http://ieeexplore.ieee.org/ Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Metadata Record: https://dspace.lboro.ac.uk/2134/5692

Version: Published

Publisher: © IEEE

Please cite the published version.
This item was submitted to Loughborough’s Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
(In)Security of Efficient Tree-Based Group Key Agreement Using Bilinear Map

Raphael C.-W. Phan\textsuperscript{1}, Member, IEEE; and Bok-Min Goi\textsuperscript{2}, Member, IEEE
\textsuperscript{1}Electronic & Electrical Engineering, Loughborough University, UK
r.phan@lboro.ac.uk
\textsuperscript{2}Centre for Multimedia Security and Signal Processing, Cryptography & Information Security Research Group, Faculty of Engineering, Multimedia University, Malaysia
bmgoi@mmu.edu.my

Abstract

A group key agreement protocol enables three or more parties to agree on a secret group key to allow for communication of secret messages between them. In this paper, we consider the security of an efficiency-improved version of the tree-based group key agreement protocol using bilinear maps proposed by Lee et al., and claimed to reduce computational costs while preserving security. To be precise, we show several attacks on this protocol and discuss how they could have been avoided.

1. Introduction

A protocol \cite{4} is a set of rules that define how communication is to be done between two or more parties. In a common networked environment where the communication channel is open to eavesdropping and modifications, security is a critical issue. In this context, security protocols are cryptographic protocols that allow communicating parties to perform mutual authentication, key exchange or both \cite{5}.

In \cite{6}, Lee et al. proposed a ternary tree-based group key agreement protocol by using bilinear maps. This builds on the tree-based Diffie-Hellman protocol (TGDH) of \cite{9,10}. In this paper, we show how the Lee et al. protocol allows both insider and outsider attackers to cause group members to compute a group key that is known to the attackers.

In Section 2, we review the group key agreement protocol proposed by Lee et al. \cite{6}. We present our attacks in Section 3. We give concluding remarks in Section 4.

1.1. Security Criteria for Protocols

A protocol is insecure when its intended security goals are not met \cite{1-3}. Among the standard security criteria \cite{8-10} for key agreement protocols are as follows:

Group key secrecy. The basic property that it is computationally infeasible for a passive attack to discover any group key.

Known-group key secrecy. Knowledge of previous group keys will not enable an attacker to know other group keys.

Perfect forward secrecy. Even when the long-term private key is compromised, it will not enable the attacker to know the values of previous group keys.

Key-compromise impersonation resilience. Even when the long-term private key is compromised, it will not enable the attacker to impersonate entities other than the owner of the private key.

Unknown key-share resilience. An attacker convinces a group of entities that they share a key with the attacker, when in fact the key is shared with another entity.

Key control resilience. It is not possible for any of the entities or the attacker to force the group key to be a pre-selected value or predict the value of the group key.

Lee et al.’s protocol appears to provide group key secrecy and known group key secrecy. Perfect forward secrecy and key-compromise impersonation resilience are not relevant here since members do not have any long-term private keys. Finally, we will show in the later sections how this protocol does not provide unknown key-share resilience and key control resilience.

In Section 2, we review the group key agreement protocol proposed by Lee et al. \cite{7}. We present our attacks in Section 3. Finally, we conclude in Section 4.
2. Tree-Based Group Key Agreement

We now briefly review the tree-based group key agreement protocol proposed by Lee et al. in [6]. This is basically a direct variant of Kim et al.’s Tree-based Group Diffie-Hellman (TGDH) [10] that replaces DH with bilinear maps and replaces binary trees with ternary trees. We first define the notations used as follows:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Number of group members</td>
</tr>
<tr>
<td>C</td>
<td>Set of current group members</td>
</tr>
<tr>
<td>L</td>
<td>Set of leaving members</td>
</tr>
<tr>
<td>M_i</td>
<td>i&lt;sup&gt;th&lt;/sup&gt; group member</td>
</tr>
<tr>
<td>h</td>
<td>Height of the key tree</td>
</tr>
<tr>
<td>&lt;l,v&gt;</td>
<td>v&lt;sup&gt;th&lt;/sup&gt; node at the l&lt;sup&gt;th&lt;/sup&gt; level in a tree</td>
</tr>
<tr>
<td>T_i</td>
<td>M_i’s view of the key tree</td>
</tr>
<tr>
<td>T_i’</td>
<td>M_i’s modified tree after membership operation</td>
</tr>
<tr>
<td>r_i</td>
<td>The key, K_{&lt;l,v&gt;}, chosen by M_i at node &lt;l,v&gt;</td>
</tr>
<tr>
<td>BK_i*</td>
<td>Set of M_i’s public blinded keys</td>
</tr>
<tr>
<td>P</td>
<td>Public information, a point on an elliptic curve</td>
</tr>
</tbody>
</table>

Figure 1 shows an example key tree. Root is at the 0<sup>th</sup> level and the lowest leaves are at the h<sup>th</sup> level. Each node, <l,v> is associated with the secret key, K_{<l,v>}, and the blinded key, BK_{<l,v>} = K_{<l,v>}P. The multiplication kP is obtained by repeating k times addition over an elliptic curve. We assume that a leaf node <l,v> knows every key along the path from <l,v> to <0,0>, called the key-path, or his view of the tree, T_i.

Figure 2 illustrates how the tree is updated while Figure 3 summaries the sequence of events.

![Tree update in join operation](image)

From Figure 3, it is clear that the new member is responsible for choosing his own secret key, K_{<l,v>} = r_{n+1} upon wishing to join the group. He starts with step 1 by computing the corresponding blinded key, bkey of r_{n+1} and sending this to all existing members of the group. The tree is then updated according to Figure 2. An existing member who is the rightmost leaf node in the sub-tree rooted at the insertion point would act as the sponsor who has to update all bkeys on the affected key-path, and broadcast them to all members.
2.2 The Leave Operation

Figure 4 similarly summarizes the steps for the leave operation when an existing member leaves the group.

Step 1: The new member broadcasts a request for join:
\[ M_{n+1} \xrightarrow{BK_{<0,0>} = r_{n+1}P} C \]

Step 2: Every member:
- if key tree contains the sub-tree that has 2 child nodes, add the new member node to update the key tree. Else, add the new member node and new intermediate node
- remove all keys and bkeys from the leaf node related to the sponsor to the root note

The sponsor, M, additionally:
- generates new share and computes all \([key, bkey]\) pairs on the key-path
- broadcasts updated \(T_s'\) containing all bkeys:
\[ M_s \xrightarrow{T_s(BK^*_s)} C \cup \{M_n+1\} \]

Step 3: Every member computes the group key using \(T_s'\).

2.3 The Merge and Partition Operations

The merge and partition operations are merely multiple rounds of the join and leave operations, respectively.

3. Attacks on the Protocol

In this section, we show that the scheme fails to meet standard security criteria, namely it does not provide unknown key-share resilience and key control resilience.

3.1 Attacks on the Join Operation

We first present attacks on the join operation. In particular, an attacker (insider or outsider) is able to force the group key to a pre-selected value. This shows that the protocol does not provide key control resilience. Attacks on other protocols will be described in the next subsection.

**Outsider Attack 1: Man-in-the-Middle (MITM) Attack on Join Request Message.** Recall that whenever a new member \(M_{n+1}\) wishes to join the group, he broadcasts a join request message to all members, \(C\), that contains its own bkey, \(BK = r_{n+1}P\). Note that although one would be unable to derive \(r_{n+1}\) from this message, however, an outside attacker could replace the entire message \(r_{n+1}P\) with another \(r'_{n+1}P\) since it provides no integrity protection! Once this is put in place, the group key computed by all members, \(C\), would contain \(r'_{n+1}\), and so this group key can also be computed by the attacker.

**Outsider Attack 2: Man-in-the-Middle (MITM) Attack on Sponsor Broadcast Message.** Recall that during the join operation, upon successful verification of a new member’s join request, the sponsor, \(M_s\), would send a broadcast message containing all newly computed bkeys, \(BK_{<n+1>}\) along the key-path [6] of the new member. Due to the same integrity problem exploited in Attack 1 above, an outside attacker could also replace this broadcast message with its own \(BK'_{<n+1>}\), thereby causing all affected members to compute new keys in the key-path that can also be computed by the attacker.

**Insider Attack:** Man-in-the-Middle (MITM) Attack on Sponsor Broadcast Message. The previous two attacks were carried out by outsiders in order to compute the group key. We also outline an attack that is useful to an insider in order to compute any key share, \(K_{<i>}\) which is not necessarily the group key. We refer to the tree \(T_{i0}\) in Figure 2 as an example. \(K_{<1,0>}\) is supposed to be known only to \(M_1\), \(M_2\), and \(M_3\). However, any other member, for instance \(M_6\), could replace either one of \(BK_{<2,0>}\), \(BK_{<3,1>}\) or \(BK_{<3,2>}\) in the sponsor broadcast message with its own. Let's suppose
he replaced $BK_{<2,0}>$ with $BK'_ {<2,0>}$. He can then compute $K_{<1,0>}$ by using $BK_{<2,1>}, BK_{<2,2>}$ (which are known to every member of the group) and $K'_ {<2,0>}$ (since this share was used by him to form $BK'_ {<2,0>} = K'_ {<2,0>}$).

3.2 Attacks on the Leave Operation

Attacks on the leave operation also follow along similar lines as those on the join operation.

**Outsider Attack:** Man-in-the-Middle (MITM) on Sponsor Broadcast Message. In the leave operation, once a leaving member has sent out a Leave message, the sponsor, $M_s$ sends a broadcast message containing all newly computed bkeys, $BK_{<n+1>}$ along the key-path of the leaving member. Due to the same integrity problem exploited in the attacks on the join operation, an attacker could replace this broadcast message with its own $BK'_{<n+1>}$, causing all affected members to compute new keys in the key-path that can also be computed by the attacker.

3.3 Extension to Merge and Partition

Since the merge and partition operations are simply combinations of several simultaneous join and leave operations respectively, our attacks on join and leave operations would equally apply to them.

3.4 Unknown Key-Share Attacks

Since this is a key agreement protocol, it should be secure against protocol-level attacks; all these should be considered as standard security criteria. However, this protocol is not secure against the unknown key-share attack.

We refer to Figure 1 as an example. Suppose $M_1$ broadcasts a join request message, $BK = r_5'P$ that is supposedly from a new member, $M_8$. Other members have no way of knowing that this join request message did not come from $M_8$ and so a new group key is established and all the key shares along the affected key path are updated. $M_1$ thinks he shares the keys $K_{<1,2>}$ with $M_8$ when in fact he is sharing it with $M_7$!

This shows that the protocol does not provide unknown key-share resilience.

4 Concluding Remarks

In this paper, we have presented attacks on the group key agreement protocol proposed by Lee et al. and shown that it fails to securely establish a group key that should be known only to legitimate group members. We remark that interestingly Kim et al.‘s protocol [10] upon which Lee et al.’s protocol is derived from, does not succumb to our attacks since the messages in Kim et al.’s protocol are signed by the senders.

5. References


