This item was submitted to [Loughborough's Research Repository](#) by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

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

PLEASE CITE THE PUBLISHED VERSION

PUBLISHER

© IEEE

VERSION

VoR (Version of Record)

LICENCE

CC BY-NC-ND 4.0

REPOSITORY RECORD

Phan, Raphael C.-W., and Bok-Min Goi. 2019. "(in)security of Efficient Tree-based Group Key Agreement Using Bilinear Map". figshare. <https://hdl.handle.net/2134/5692>.

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. Phan¹, Member, IEEE; and Bok-Min Goi², Member, IEEE

¹*Electronic & Electrical Engineering, Loughborough University, UK*
r.phan@lboro.ac.uk

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 [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 [5].

In [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 [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. [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 [1-3]. Among the standard security criteria [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 an 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. [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:

N	Number of group members
C	Set of current group members
L	Set of leaving members
M_i	i^{th} group member
h	Height of the key tree
$\langle l, v \rangle$	v^{th} node at the l^{th} level in a tree
T_i	M_i 's view of the key tree
T_i'	M_i 's modified tree after membership operation
r_i	The key, $K_{\langle l, v \rangle}$ chosen by M_i at node $\langle l, v \rangle$
BK_i^*	Set of M_i 's public blinded keys
P	Public information, a point on an elliptic curve

Figure 1 shows an example key tree. Root is at the 0^{th} level and the lowest leaves are at the h^{th} level. Each node, $\langle l, v \rangle$ is associated with the secret key, $K_{\langle l, v \rangle}$ and the blinded key, $BK_{\langle l, v \rangle} = K_{\langle l, v \rangle}P$. The multiplication kP is obtained by repeating k times addition over an elliptic curve. We assume that a leaf node $\langle l, v \rangle$ is associated with M_i , then the node $\langle l, v \rangle$ has M_i 's session random key, $K_{\langle l, v \rangle}$. We further assume that the member M_i at node $\langle l, v \rangle$ knows every key along the path from $\langle l, v \rangle$ to $\langle 0, 0 \rangle$, called the *key-path*, or his view of the tree, T_i .

Fig.1. A key tree example

For example, member M_3 owns the tree T_3 so he knows every key $\{K_{\langle 2,2 \rangle}, K_{\langle 1,0 \rangle}, K_{\langle 0,0 \rangle}\}$ and every *bkey*, $BK_3^* = \{BK_{\langle 2,2 \rangle}, BK_{\langle 1,0 \rangle}, BK_{\langle 0,0 \rangle}\}$ on T_3 . Any key, $K_{\langle l, v \rangle}$ can

be computed with the knowledge of the *key* in one of its three child nodes and the *bkeys* of the other child nodes. For example in Figure 1, $K_{\langle 1,1 \rangle}$ can be computed with the knowledge of $K_{\langle 2,3 \rangle}$, $BK_{\langle 2,4 \rangle}$ and $BK_{\langle 2,5 \rangle}$, or $BK_{\langle 2,3 \rangle}$, $K_{\langle 2,4 \rangle}$ and $BK_{\langle 2,5 \rangle}$, or $BK_{\langle 2,3 \rangle}$, $BK_{\langle 2,4 \rangle}$ and $K_{\langle 2,5 \rangle}$. Since our attacks do not exploit how the *key* is computed, we will omit the description here and instead refer the interested reader to [6] for further details. The protocol consists of the *join*, *leave*, *merge* and *partition* operations.

2.1. The Join Operation

This operation specifies what happens and how the group keys are updated when a new member, M_{n+1} joins the group. Figure 2 illustrates how the tree is updated while Figure 3 summaries the sequence of events.

Fig.2. Tree update in join operation

From Figure 3, it is clear that the new member is responsible for choosing his own secret key, $K_{\langle l, v \rangle} = r_{n+1}$ 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.

Fig.3. Join operation

Fig. 4. Leave operation

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*, $BK_{\langle n+1 \rangle}$ 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 $BK'_{\langle n+1 \rangle}$, 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, $K_{\langle i \rangle}$ which is not necessarily the group key. We refer to the tree T_{10} in Figure 2 as an example. $K_{\langle 1,0 \rangle}$ 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 $BK_{\langle 2,0 \rangle}$, $BK_{\langle 2,1 \rangle}$ or $BK_{\langle 2,2 \rangle}$ in the sponsor broadcast message with its own. Lets suppose

he replaced $BK_{\langle 2,0 \rangle}$ with $BK'_{\langle 2,0 \rangle}$. He can then compute $K_{\langle 1,0 \rangle}$ by using $BK_{\langle 2,1 \rangle}$, $BK_{\langle 2,2 \rangle}$ (which are known to every member of the group) and $K'_{\langle 2,0 \rangle}$ (since this share was used by him to form $BK'_{\langle 2,0 \rangle} = K'_{\langle 2,0 \rangle}P$).

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*, $BK_{\langle n+1 \rangle}$ 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 $BK'_{\langle n+1 \rangle}$, 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, $BK = r_8'P$ 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_7 thinks he shares the keys $K_{\langle 1,2 \rangle}$ with M_8 when in fact he is sharing it with M_1 ! 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

- [1] M. Abadi, "Explicit Communication Revisited: Two New Attacks on Authentication Protocols," IEEE Transactions on Software Engineering, vol. 23, no. 3, pp. 185-186, 1997.
- [2] G. Lowe, "An attack on the Needham-Schroeder public-key protocol," Information Processing Letters, vol. 56, pp. 131-133, 1995.
- [3] G. Lowe, "Some New Attacks upon Security Protocols," Proceedings of 9th IEEE Computer Security Foundations Workshop, 1996.
- [4] R. M. Needham and M. D. Schroeder, "Using encryption for authentication of large networks and computers", Communications of the ACM, vol. 21, no. 12, pp. 993-999, 1978.
- [5] R. M. Needham and M. D. Schroeder, "Authentication revisited", Operating Systems Review, vol. 21, no. 7, 1987.
- [6] S. Lee, Y. Kim, K. Kim and D.-H. Ryu, "An Efficient Tree-Based Group Key Agreement Using Bilinear Map", Proceedings of the Conference on Applied Cryptography and Network Security (ACNS '03), LNCS, vol. 2846, Springer-Verlag, pp. 357-371, 2003.
- [7] P. Syverson, "A Taxonomy of Replay Attacks", Proceedings of the 7th IEEE Computer Security Foundations Workshop, pp. 131-136, 1994.
- [8] Y. Kim, A. Perrig and G. Tsudik, "Communication-Efficient Group Key Agreement", Proceedings of the IFIP SEC '01, pp. 229-244, 2001.
- [9] Y. Kim, A. Perrig and G. Tsudik, "Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups", Proceedings of the ACM-CCS '00, pp. 235-244, 2000.
- [10] Y. Kim, A. Perrig and G. Tsudik, "Tree-Based Group Diffie-Hellman Protocol", IACR Cryptology ePrint Archive, Report 2003/122, Available online at <http://eprint.iacr.org/2002/009/>.