Pseudonym Generation Scheme for Ad-Hoc Group Communication Based on IDH

Mark Manulis and Jörg Schwenk

Network and Data Security Group,
Department of Electrical Engineering & Information Sciences,
IC 4/158, Ruhr-Universität Bochum, D-44801, Germany
{mark.manulis, joerg.schwenk}@rub.de

Abstract. In this paper we describe the advantages of using iterative Diffie-Hellman (IDH) key trees for mobile ad-hoc group communication. We focus on the Tree-based Group Diffie-Hellman (TGDH) protocol suite, that consists of group key agreement protocols based on IDH key trees. Furthermore, we consider the anonymity of members during group communication over a public broadcast channel that provides untraceability of messages. The main goal of the proposed pseudonym generation scheme is to allow group members to generate their own pseudonyms that can be linked to their real identities only by a democratic decision of some interacting group members. The real identities are bound to public keys used in the group key agreement. The communication and computation costs as well as the security of the scheme can be optimized with respect to the characteristics of involved mobile devices.

1 Introduction

In this paper we describe the communication and computation advantages of the iterative Diffie-Hellman (IDH) key agreement for ad-hoc group communication scenarios and propose a new pseudonym generation scheme with threshold revocation which can be embedded in the IDH process. We are using IDH key agreement protocols of the Tree-based Group Diffie-Hellman (TGDH) suite, proposed recently in [7]. As communication infrastructure we consider a public broadcast channel between mobile devices that provides untraceability of messages, where only the attached identity values serve as identification of the sender. In our scenario members agree on a group key using the TGDH protocols and then compute their own pseudonyms using our pseudonym generation scheme, which is an extension to TGDH. These pseudonyms are unlinkable to the real identities of members used in the group key agreement. The real identity can be revealed only upon interaction of a certain number of group members in a democratic decision. Besides that, we propose a new communication and computation costs optimizing strategy given by the structure of the key trees established by the iterative Diffie-Hellman key agreement. It allows to choose the optimization cost factor that is common for all mobile devices that take part in ad-hoc communication. The strategy is based on the fact that keys assigned to IDH-tree nodes are shared between members assigned to leaves of the subtree rooted at that node. The optimization of costs is very

C. Castelluccia et al. (Eds.): ESAS 2004, LNCS 3313, pp. 107–124, 2005.

important for ad-hoc communication, since mobile devices are often limited in their power ressources.

Motivation. Different appliance scenarios can be considered for the ad-hoc group communication with pseudonyms, e. g. members of directing board of a company might want to communicate securely and anonymously in order to perform an anonymous election process, without having to trust into a third party. If at least one of members breaches the communication rules by broadcasting some misleading information, then other members might want to reveal her identity. The decision whether such dispute case has been occured is democratic since none of group members is obliged to take part in the revealing process. This is the main difference to communication scenarios with a designated group manager that decides when dispute case has occured. To achieve such democratic decision our scheme allows a subset of k members to trace any pseudonym to its holder, with k being a power of 2. Another example is an ad-hoc analogon to the GSM TMSI (Temporary Mobile Subscriber Identity) that would allow users to hide the real identity of their mobile devices by generating pseudonyms. In this case users are interested in generating such pseudonyms, since otherwise it would be possible to track their mobile devices.

Organization. The rest of the paper is organized as follows. Section 2 describes the IDH key agreement protocols of the TGDH suite. Section 3 outlines computation and communication costs of the TGDH protocols and shows their advantages in a mobile ad-hoc environment. Section 4 specifies the communication model and requirements defined for the proposed pseudonym generation scheme, that is presented in Section 5. We describe first a special case, when all group members have to interact in order to link a pseudonym to its real identity. Further, we give a generalized pseudonym generation scheme, that uses the structure of the IDH key tree, and mention additional optimizations for the limited power ressources of the mobile devices.

2 Iterative Diffie-Hellman (IDH) Key Agreement

Iterative Diffie-Hellman key agreement was originally proposed in [2] and later more specific in [5] and [7]. In [7] Kim *et. al.* introduce the Tree-based Group Diffie-Hellman (TGDH) protocol suite which allows group members to establish and maintain a group key through a *contributory* agreement, where each member contributes her own share to the common group key. There is no group manager or any other trusted authority required for this agreement. In the following, we give a brief description of the main protocols of the TGDH suite.

2.1 IDH Key Tree

The IDH key tree used in the TGDH protocol suite is a logical binary tree, referred to as $\mathcal{T}$. It consists of nodes $\langle l, v\rangle$, the v-th node at level l, $0 \leq v \leq 2^l - 1$ and $0 \leq l \leq h$ where h is the height of $\mathcal{T}$. Group members are represented by leaf nodes. Node $\langle 0, 0\rangle$ is the root of $\mathcal{T}$. Each node $\langle l, v\rangle$ is associated with a key $K_{\langle l,v\rangle}$ and a blinded key

(bkey) $BK_{\langle l,v\rangle} = f(K_{\langle l,v\rangle})$, where f is an exponentiation function $f(k) = g^k$ in a multiplicative cyclic prime order group $\mathbb{G}$ with generator g. If we assume that leaf node $\langle l, v\rangle$ hosts member M, then M knows the keys of all nodes on the path from node $\langle l, v\rangle$ to $\langle 0, 0\rangle$, referred to as the *key path*. Keys associated with leaves are chosen by the group members. The keys $K_{\langle l,v\rangle}$ in $\mathcal{T}$, where $\langle l, v\rangle$ is not a leave, are computed iteratively using the Diffie-Hellman key exchange as follows:

$$\begin{aligned} K_{\langle l,v\rangle} &= (BK_{\langle l+1,2v+1\rangle})^{K_{\langle l+1,2v\rangle}} \\ &= (BK_{\langle l+1,2v\rangle})^{K_{\langle l+1,2v+1\rangle}} \\ &= g^{K_{\langle l+1,2v\rangle} K_{\langle l+1,2v+1\rangle}} \\ &= f(K_{\langle l+1,2v\rangle} K_{\langle l+1,2v+1\rangle}) \end{aligned}$$

Member M can compute key $K_{\langle l,v\rangle}$ of any node $\langle l, v\rangle$ on her key path only if she knows the key of its preceeding node $\langle l + 1, 2v\rangle$ or $\langle l + 1, 2v + 1\rangle$ and the bkey of the sibling node of that preceeding node ($\langle l + 1, 2v + 1\rangle$ or $\langle l + 1, 2v\rangle$, respectively). Every blinded key $BK_{\langle l,v\rangle}$ is sent over the broadcast channel after being computed by one of the group members in the subtree of a node. Every member M knows her position in $\mathcal{T}$ and can therefore determine which blinded keys are needed to compute the keys on her key path. The computation of a key path is finished with the calculation of the group key $K_{\langle 0,0\rangle}$. For security reasons $K_{\langle 0,0\rangle}$ should not be used directly for the purpose of encryption, authentication or data integrity, but some derived key (e.g. $K_{group} = h(K_{\langle 0,0\rangle})$, where h is a strong hash function).

For example, in Fig. 1, M_2 can compute $K_{\langle 2,0\rangle}$, $K_{\langle 1,0\rangle}$ and $K_{\langle 0,0\rangle}$ using $K_{\langle 3,1\rangle}$, $BK_{\langle 3,0\rangle}$, $BK_{\langle 2,1\rangle}$ and $BK_{\langle 1,1\rangle}$.

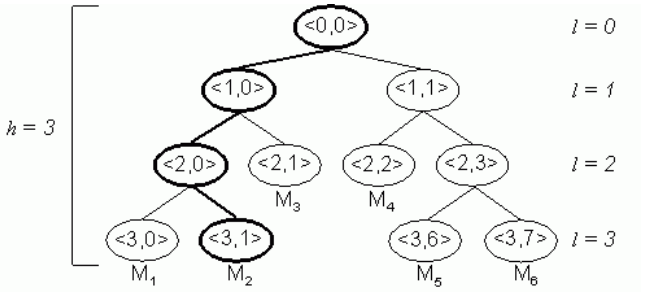

Fig. 1. IDH Key Tree of TGDH Protocol Suite

To establish and maintain such an IDH key tree the TGDH suite provides JOIN, LEAVE, PARTITION and MERGE protocols. Due to space limitations we describe only JOIN and LEAVE.

2.2 TGDH Protocol Suite: JOIN

For a set of group members $\mathcal{M} = \{M_1, \ldots, M_n\}$ and a new member M_{n+1} Fig. 2 describes the JOIN protocol of the TGDH suite. The insertion point of the new member

1. M_{n+1} chooses her own key K_{n+1}, computes bkey BK_{n+1} and broadcasts the latter as request for join
2. Every member M_i, $1 \leq i \leq n$
   - calculates the insertion point of M_{n+1} in $\mathcal{T}$
   - updates $\mathcal{T}$ by adding two leaf nodes for M_{n+1} and for the sponsor at the insertion point node
   - removes (invalidates) all keys and bkeys along the key path of the new leaf nodes

   The *sponsor* M_s additionally
   - computes all (key, bkey) pairs on her key path
   - broadcasts an updated tree $\hat{\mathcal{T}}$ including bkeys of all its nodes
3. Every member M_i, $1 \leq i \leq n+1$ updates her copy of $\mathcal{T}$ with the new bkeys and computes new keys for all the nodes of her key path intersecting the sponsor node key path, including the group key $K_{\langle 0,0 \rangle}$

Fig. 2. TGDH JOIN Protocol

is the rightmost node, where (if possible) JOIN does not increase the height of $\mathcal{T}$ (and else in a completely balanced tree simply the rightmost node). The sponsor is the group member initially located at the insertion point; he will move to the left leave node which will be added to $\mathcal{T}$ during the JOIN operation. The right leave node will be inhabited by the new group member. The sponsor shares the first Diffie-Hellman key with the new member, updates the tree and broadcasts the changed blinded key(s).

Fig. 3 shows an example of member M_4 joining a group $\mathcal{M} = \{M_1, M_2, M_3\}$ at insertion point $\langle 1,1 \rangle$. Member M_3 is hosted by node $\langle 1,1 \rangle$ and is therefore the sponsor. She renames her leaf node $\langle 1,1 \rangle$ to $\langle 2,2 \rangle$, then adds new intermediate node $\langle 1,1 \rangle$ and new member's leaf node $\langle 2,3 \rangle$ to $\mathcal{T}$, computes keys $K_{\langle 1,1 \rangle}$ and $K_{\langle 0,0 \rangle}$ on her key path using her own key $K_{\langle 2,2 \rangle}$ and bkeys $BK_{\langle 2,3 \rangle}$ and $BK_{\langle 1,0 \rangle}$. After M_3 broadcasts the updated tree $\hat{\mathcal{T}}$ every member can compute keys on her key path too.

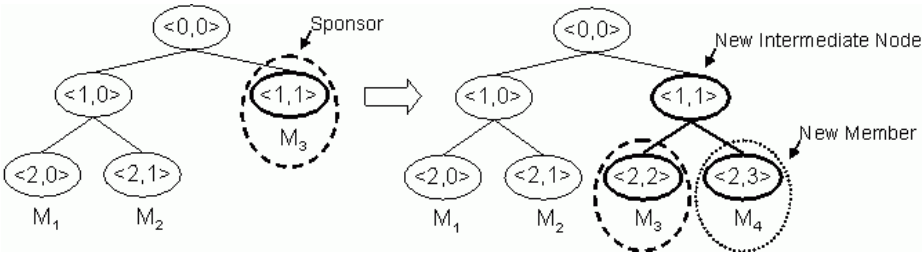

Fig. 3. Example of JOIN Protocol

2.3 TGDH Protocol Suite: LEAVE

For a set of group members $\mathcal{M} = \{M_1, \ldots, M_n\}$ let M_d be a member who leaves the group. Fig. 4 shows the LEAVE protocol of the suite. The sponsor is the rightmost leaf node of the subtree rooted at the sibling node of M_d. Fig. 5 shows an example of member M_3 leaving the group. Every remaining member deletes nodes $\langle 2,2 \rangle$ and $\langle 1,1 \rangle$ from $\mathcal{T}$.Sponsor M_5 computes changed keys $K_{\langle 1,1 \rangle}$ and $K_{\langle 0,0 \rangle}$ on her key path using her own

1. Every member M_i, $1 \leq i \leq n$, $i \neq d$
   - updates $\mathcal{T}$ by removing the member node of M_d and replacing its parent node with the sibling node of M_d
   - removes (invalidates) all keys and bkeys along the key path of the leaving member node

   The *sponsor* M_s additionally
   - computes all (key, bkey) pairs on her key path
   - broadcasts an updated tree $\hat{\mathcal{T}}$ including bkeys of all its nodes
2. Every member M_i, $1 \leq i \leq n-1$ updates her copy of $\mathcal{T}$ with the new bkeys and computes new keys for all the nodes of her key path intersecting the sponsor node key path, including the group key $K_{\langle 0,0 \rangle}$

Fig. 4. TGDH LEAVE Protocol

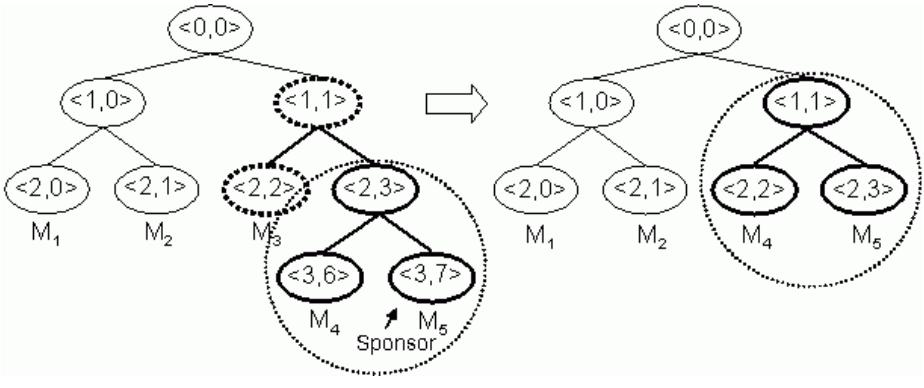

Fig. 5. Example of LEAVE Protocol

key $K_{\langle 2,3 \rangle}$ and bkeys $BK_{\langle 2,2 \rangle}$ and $BK_{\langle 1,0 \rangle}$. After M_5 broadcasts the updated tree $\hat{\mathcal{T}}$ every remaining member can compute keys on her key path too.

3 IDH Key Agreement in Ad-Hoc Networks

Let us summarize some properties of the TGDH protocols that make them suitable for mobile ad-hoc group communication scenarios. When designing protocols for wireless ad-hoc communication, several factors should be considered.

- The relationship between communication participants in mobile networks is highly dynamical and temporary. Communication can be interrupted at any time due to the mobility of nodes. Especially for secure group communication in such environments fault tolerance and efficient dynamic maintenance are required for the group key agreement.
- Another property of mobile ad-hoc networks is the absence of trusted nodes. Any node can be a potential adversary. Therefore a decentralized and cooperative participation of all nodes having equal rights must be provided to achieve desired security for the group key establishment.

– Mobile stations are limited in their computational capability compared to other stationary devices.

Protocols of the TGDH suite fulfill the described requirements. The dynamical and temporary nature of the interaction of group members prohibits the use of symmetric key cryptography, which needs a secure channel to establish cryptographic keys. Amongst the published solutions for public key group key agreement schemes, the TGDH protocols offer the best performance for JOIN and LEAVE operations. Computation and communication costs of the specified protocols are shown in Table 1. The height of the current IDH key tree, number of merging groups and leaving members are denoted by: h, k and p, respectively.

Table 1. Computation and Communication Costs of the TGDH Protocols

	Computation			Communication	
	Exponentiations	Signatures	Verifications	Rounds	Messages
JOIN	$\frac{3h}{2}$	2	3	2	3
LEAVE	$\frac{3h}{2}$	1	1	1	1
MERGE	$\frac{3h}{2}$	$\log_2 k + 1$	$\log_2 k$	$\log_2 k + 1$	$2k$
PARTITION	$3h$	$min(\log_2 p, h)$	$min(\log_2 p, h)$	$min(\log_2 p, h)$	$2h$

Kim *et. al.* [7] compare the TGDH protocols to other existing contributory group key agreement protocols like GDH.3 [8], BD (Burmester-Desmedt) [3], and STR [6]. Despite the higher cost of partition, the TGDH suite shows the best performance on low and medium delay networks. In [1] is shown, how delay in wireless ad-hoc networks can be kept low without trading off the throughput while capacity of a network increases. This confirms the possibility of efficient usage of the TGDH protocols for wireless ad-hoc communication. Mobile peers with limited power can perform the protocols due to their low computation costs. The contributory nature of the protocols allows to compute the group key in a decentralized and cooperative manner without trusted parties. The sponsor in the protocols is not a privileged entity. She is chosen according to the actual tree structure and her messages can be verified by other participants, so that no compromisation of the key agreement is possible. Additionally, we outline some points about fault tolerance.

Suppose that a new member M_{n+1} wants to join the group and looses her connection after sending her blinded key BK_{n+1} in step 1 of the JOIN protocol. Despite that, remaining group members can still perform steps 2 and 3 and agree on a new group key. If M_{n+1} restores her connection to the group, she would only need to verify the knowledge of her corresponding own key K_{n+1}. After receiving the updated tree, she can compute the group key according to step 3 of the protocol without additional interaction.

Network failures that lead to more partitions of the group can be managed using the PARTITION protocol. With the MERGE protocol different groups can be efficiently merged to a common group.

4 Communication Model and Problem Specification

In this section we specify the communication model and define some requirements for our pseudonym generation scheme.

COMMUNICATION MODEL: A set $\mathcal{M} = \{M_1, \ldots, M_n\}$ of n network participants, called *members*, communicates over a public broadcast channel, that provides untraceability of messages. The sender of the message can only be identified upon some identity value id attached to a message.

We consider that members communicate in order to form a dynamic group whithout having any privileged members or trusted parties. Especially, there is no group manager with extended rights. Each member has a unique real identity, which she attaches to messages of the key agreement protocol. Our main goal is to give members pseudonyms, that can be used as id values in messages in order to hide the real identities once the group key is established. Pseudonyms should fulfil the following requirements:

- Each pseudonym can be linked to the real identity of M_i only through the efficient interaction of k members, where k is a fixed value
- The pseudonym of any member M_i must be computed efficiently with respect to the given computational and communicational ressources
- Any member must be able to choose her own pseudonym

5 Pseudonym Generation Scheme Based on IDH Key Trees

5.1 Pseudonym Generation Scheme (Special Case $k = n$)

Our main aim is to embed pseudonym generation in the IDH key agreement. Therefore computations of our scheme are done in the same group $\mathbb{G}$, that is used by the TGDH protocols. Recall that $\mathbb{G}$ is a multiplicative cyclic prime order group with generator g and keys of the IDH key tree are computed using some exponentiation function $f(k) = g^k$.

Remark 1. We remark at this point, that our scheme works also in any other cyclic group, where the *Computation Diffie-Hellman Problem* is hard. Therefore $\mathbb{G}$ can also be a group of points of an elliptic curve E over a finite field $\mathbb{F}_p$ with prime p or $\mathbb{F}_{2^m}$. Such groups are appropriate for use in mobile ad-hoc networks because of the low computation costs of the group operations.

First we describe how our pseudonym generation works using a given IDH key tree $\mathcal{T}$ for a set of current group members $\mathcal{M} = \{M_1, \ldots, M_n\}$. Pseudonyms generated by our scheme can be linked to the real identities only upon the interaction of k members. In this section we describe a special case of $k = n$. In Section 5.2 we then show how k and computational costs can be optimized with respect to the limited ressources of given ad-hoc devices using the structure of the IDH key tree.

As mentioned in Section 2 a member M is represented by a leaf node $\langle l, v \rangle$ with private key $K_{\langle l,v \rangle}$ and blinded key $BK_{\langle l,v \rangle}$. For simplicity we denote them as K_i and

BK_i respectively, for a member $M_i \in \mathcal{M}, i \in \{1, \ldots, n\}$. Blinded keys BK_i are public and computed as

$$BK_i = g^{K_i}$$

Let M_s perform our pseudonym scheme. M_s computes the *shared keys*

$$skey_{sj} = BK_j^{K_s} = g^{K_j K_s}$$

for $j \in \{1, \ldots, n\}$, $j \neq s$. For a chosen pseudonym ps_s and each shared key $skey_{sj}$, M_s constructs the *secret share*

$$s_{sj} = skey_{sj}^{ps_s} = g^{K_j K_s ps_s}$$

and uses it to compute the *public share*

$$S_s = g^{\sum_j s_{sj}} = g^{\sum_j g^{K_j K_s ps_s}}$$

M_s broadcasts the public share and uses pseudonym ps_s as her identity value id in her further group messages.

Remark 2. Although, our scheme allows any value to be chosen for the pseudonym ps, a better choice is to chose some secret random value xs and compute $ps = g^{xs}$. This would allow the holder of the pseudonym to use (xs, ps) as a public-key pair during the communication, e. g. to sign or encrypt messages using own pseudonym.

In the following, we describe how all n group members interact to link the pseudonym ps_s to its holder.

Every member M_i computes secret shares

$$s_{ij} = skey_{ij}^{ps_s} = g^{K_j K_i ps_s}$$

for $j \in \{1, \ldots, n\}$, $j \neq i$ and broadcasts *hidden secret shares*

$$r_{ij} = g^{s_{ij}} = g^{g^{K_j K_i ps_s}}$$

M_i can verify whether pseudonym ps_s is linked to member M_s by computing

$$R_s = \prod_j r_{js} = \prod_j g^{g^{K_s K_j ps_s}}$$

using hidden secret shares r_{js}, $j \in \{1, \ldots, n\}$, $j \neq s$ and comparing the product to the public share S_s of M_s.

Theorem 1 (Special Case $k = n$). *If pseudonym ps_s was correctly used in the construction of S_s by member M_s then $R_s = S_s$.*

Proof.

$$S_s = g^{\sum_j s_{sj}} = g^{\sum_j g^{K_s K_j ps_s}} = \prod_j g^{g^{K_s K_j ps_s}} = \prod_j r_{js} = R_s$$

□

Member M_i does not know in advance who owns the pseudonym ps_s. Therefore, the above computations should be done for all members simultaneously.

5.2 Generalized Pseudonym Generation Scheme

Our first requirement on a pseudonym in Section 4 is, that it can be linked to the owner only by interaction of k members, where k is fixed. So far, we have presented the special case $k = n$, that is all group members must exchange their hidden secret shares in order to link a pseudonym. In the following we give the generalized form of the scheme. Recall from Section 2, that members are hosted by leafs of the IDH key tree $\mathcal{T}$. Using the structure of $\mathcal{T}$, k can be varied by choosing different levels for the public keys to be used in the construction of secret shares. Let M_s perform the generalized scheme for a chosen level l and pseudonym ps_s. M_s first computes the shared keys

$$skey_{s\langle l,v\rangle} = (BK_{\langle l,v\rangle})^{K_s} = g^{K_{\langle l,v\rangle}K_s}$$

for $0 \leq v \leq 2^l - 1$ and the corresponding secret shares

$$s_{s\langle l,v\rangle} = skey^{ps_s}_{s\langle l,v\rangle} = g^{K_{\langle l,v\rangle}K_s ps_s}$$

and then builds her public share

$$S_s = g^{\sum_v s_{s\langle l,v\rangle}} = g^{\sum_v g^{K_{\langle l,v\rangle}K_s ps_s}}$$

and broadcasts it. The linking process requires the interaction of k members, referred to as set $\mathcal{K} \subseteq \mathcal{M}$. $\mathcal{K}$ should consist of at least one representative member from all subgroups, that are rooted at nodes $\langle l, v\rangle$, $0 \leq v \leq 2^l - 1$, so that there is at least one member, who knows the corresponding private key $K_{\langle l,v\rangle}$ of each secret share. Set $\mathcal{K}$ can be generally defined as

$$\mathcal{K} = \{\{M_{K_{\langle l,0\rangle}}, \ldots, M_{K_{\langle l,2^l-1\rangle}}\} \mid M_{K_{\langle l,v\rangle}} \in \mathcal{M} \text{ is a member of a subgroup rooted at node } \langle l, v\rangle\}$$

The following dependency is given between $k = |\mathcal{K}|$ and chosen level l:

$$k \leq 2^l\,, \quad 0 \leq l \leq h$$

In order to link pseudonym ps_s to its owner M_s, every member $M_{K_{\langle l,v\rangle}} \in \mathcal{K}$ must compute secret shares using private key $K_{\langle l,v\rangle}$ and public key BK_i of every other member $M_i \in \mathcal{M}$ as follows

$$s_{\langle l,v\rangle i} = skey^{ps_s}_{\langle l,v\rangle i} = BK_i^{K_{\langle l,v\rangle}ps_s} = g^{K_i K_{\langle l,v\rangle}ps_s}$$

and broadcast the hidden secret shares

$$r_{\langle l,v\rangle i} = g^{s_{\langle l,v\rangle}i} = g^{g^{K_i K_{\langle l,v\rangle}ps_s}}$$

for $0 \leq v \leq 2^l - 1$. Any member $M_i \in \mathcal{M}$ can prove whether pseudonym ps_s is owned by member M_s upon computing the product

$$R_s = \prod_v r_{\langle l,v\rangle s} = \prod_v g^{g^{K_s K_{\langle l,v\rangle}ps_s}}$$

using hidden secret shares $r_{\langle l,v\rangle s}$, $0 \leq v \leq 2^l - 1$ and comparing it with the public share S_s of M_s.

Theorem 2 (Generalized Case $k \leq 2^l$). *If pseudonym ps_s was correctly used in the computation of S_s by member M_s, then $R_s = S_s$.*

Proof.

$$S_s = g^{\sum_v s_{s\langle l,v\rangle}} = g^{\sum_v g^{K_{\langle l,v\rangle} K_s ps_s}} = \prod_v g^{g^{K_s K_{\langle l,v\rangle} ps_s}} = \prod_v r_{\langle l,v\rangle s} = R_s$$

□

The generalized pseudonym scheme is useful in ad-hoc networks, because the choice of level l and that is of k can be used to optimize costs, with respect to limited ressources of given mobile devices. In order to keep down computation costs, a lower level l should be chosen. There is more on computation and communication costs in Section 5.5.

Fig. 6 shows an example of the IDH key tree $\mathcal{T}$ for a set of five members $\mathcal{M} = \{M_1, M_2, M_3, M_4, M_5\}$. Assume, that with respect to their computation ressources members

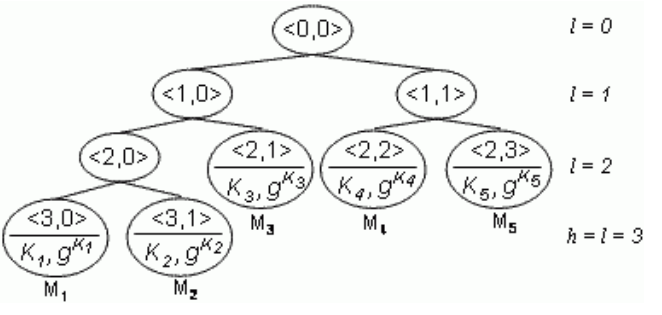

Fig. 6. Example: $\mathcal{T}$ ready for Pseudonym Generation

agree for pseudonyms generation to use public keys of level 1, that are $BK_{\langle 1,0\rangle}$ and $BK_{\langle 1,1\rangle}$. Public shares S_i are computed and broadcasted by members according to Table 2. Pseudonym ps_i can then be used as identity value for the messages of M_i.

To link pseudonym ps_i set $\mathcal{K}$ should consist of at least 2 members. One of them should know $K_{\langle 1,0\rangle}$ and another $K_{\langle 1,1\rangle}$. Set $\mathcal{K}$ is defined as

$$\mathcal{K} = \{\{M_{K_{\langle 1,0\rangle}}, M_{K_{\langle 1,1\rangle}}\} | M_{K_{\langle 1,0\rangle}} \in \{M_1, M_2, M_3\}, M_{K_{\langle 1,1\rangle}} \in \{M_4, M_5\}\}$$

We assume that $\mathcal{K} = \{M_1, M_4\}$ and pseudonym ps_2 should be linked to its owner. As shown in Table 3, members of $\mathcal{K}$ compute and broadcast hidden secret shares $r_{\langle 1,v\rangle i}$ for $v = \{0, 1\}$.

Members M_1, M_2, and M_3 belong to the subgroup routed at node $\langle 1, 0\rangle$, thus they all can compute hidden secret shares $r_{\langle 1,0\rangle i}$ for every $M_i \in \mathcal{M}$. Therefore, if M_1 sends falsified hidden secret shares, this would be noticed by other subgroup members. For the

Table 2. Example: Computation of public share S_i

Member $M_i \in \mathcal{M}$	Public share S_i
M_1	$g^{BK_{\langle 1,0\rangle}^{K_1 ps_1} + BK_{\langle 1,1\rangle}^{K_1 ps_1}}$
M_2	$g^{BK_{\langle 1,0\rangle}^{K_2 ps_2} + BK_{\langle 1,1\rangle}^{K_2 ps_2}}$
M_3	$g^{BK_{\langle 1,0\rangle}^{K_3 ps_3} + BK_{\langle 1,1\rangle}^{K_3 ps_3}}$
M_4	$g^{BK_{\langle 1,0\rangle}^{K_4 ps_4} + BK_{\langle 1,1\rangle}^{K_4 ps_4}}$
M_5	$g^{BK_{\langle 1,0\rangle}^{K_5 ps_5} + BK_{\langle 1,1\rangle}^{K_5 ps_5}}$

Table 3. Example: Broadcasted hidden secret shares $r_{\langle 1,v\rangle i}$

Member $M_i \in \mathcal{K}$	$r_{\langle 1,v\rangle 1}$	$r_{\langle 1,v\rangle 2}$	$r_{\langle 1,v\rangle 3}$	$r_{\langle 1,v\rangle 4}$	$r_{\langle 1,v\rangle 5}$
M_1	$g^{BK_1^{K_{\langle 1,0\rangle} ps_2}}$	$g^{BK_2^{K_{\langle 1,0\rangle} ps_2}}$	$g^{BK_3^{K_{\langle 1,0\rangle} ps_2}}$	$g^{BK_4^{K_{\langle 1,0\rangle} ps_2}}$	$g^{BK_5^{K_{\langle 1,0\rangle} ps_2}}$
M_4	$g^{BK_1^{K_{\langle 1,1\rangle} ps_2}}$	$g^{BK_2^{K_{\langle 1,1\rangle} ps_2}}$	$g^{BK_3^{K_{\langle 1,1\rangle} ps_2}}$	$g^{BK_4^{K_{\langle 1,1\rangle} ps_2}}$	$g^{BK_5^{K_{\langle 1,1\rangle} ps_2}}$

same reason members M_4 and M_5, that belong to the subgroup rooted at node $\langle 1,1\rangle$, can prove hidden secret shares being broadcasted by each other.

For computation of the product R_i every member $M_j \in \mathcal{M}$ has to multiply broadcasted hidden secret shares $r_{\langle 1,0\rangle i}$ and $r_{\langle 1,1\rangle i}$. After M_j builds the products R_1, R_2, R_3, R_4, and R_5, she compares them to public shares. Comparing $R_2 = S_2$, M_j can figure out, that member M_2 owns the pseudonym ps_2.

5.3 Embedding the Pseudonym Generation Scheme in TGDH

In this section we show how the generalized pseudonym scheme can be embedded in group key agreement protocols of the TGDH suite. Every new member generates her pseudonym upon joining the group. Every former group member should change her pseudonym, otherwise the pseudonym of the new member can be easily figured out. We change the last point of step 2 and extend step 3 of the original JOIN protocol of Section 2 to be conform with our pseudonym generation scheme, as described on Fig. 7. Our modifications are marked bold. Our protocols do not specify how to agree on value k in advance, however one possibility is to calculate an optimal value k for each participating mobile device and then to choose the lowest in order to allow the least powerful device to take part in communication. Another possibility is to publish optimal values for k and corresponding mobile device properties so that members can look up and set values in advance. Step 4 prevents any member from having more than one pseudonym and specifies which pseudonyms are valid for communication. If less or more than $n+1$ pseudonyms are received, pseudonym generation can be repeated without changing the group key. A member, who permanently compromises the generation, can be figured out upon revealing pseudonyms of all members. The compromising member can then be excluded from the group communication.

2. ...
   The *sponsor* M_s additionally
   - computes all (key, bkey) pairs on her key path
   - broadcasts an updated tree $\hat{\mathcal{T}}$ including bkeys of all its nodes
3. Every member M_i, $1 \leq i \leq n+1$
   - updates her copy of $\mathcal{T}$ and computes new group key $K_{\langle 0,0 \rangle}$
   - **chooses new pseudonym ps_i, computes new secret shares $s_{\langle l,v \rangle i}$ and broadcasts public share S_i with attached real identity**
   - **builds keyed-hashing value of ps_i using new group key $K_{\langle 0,0 \rangle}$ and broadcasts it anonymously**
4. **Every member verifies received hash values and ensures that there are $n+1$ pseudonyms**

Fig. 7. GENERATION Protocol as Extension to TGDH JOIN Protocol

1. Every member $M_{K_{\langle l,v \rangle}} \in \mathcal{K}$
   - computes n hidden secret shares $r_{\langle l,v \rangle i}$, using ps
   - combines all $r_{\langle l,v \rangle i}$ to a single message and builds its keyed-hashing value using $K_{\langle 0,0 \rangle}$
   - broadcasts the message and its keyed-hashing value with attached real identity
2. Every member $M_s \in \mathcal{M}$
   - verifies keyed-hashing values of received messages and extracts values $r_{\langle l,v \rangle i}$
   - computes product R_i for every other member $M_i \in \mathcal{M}$, $i \neq s$
   - finds public share $S_i = R_i$ and determines member M_i as owner of ps

Fig. 8. Basic LINKING Protocol

In order to link pseudonym ps to its holder, members perform the Basic LINKING protocol from Fig. 8. This protocol uses set $\mathcal{K}$, that consists only of one member of each subgroup rooted at node $\langle l, v \rangle$, which public key was used in the computation of the secret share $s_{\langle l,v \rangle i}$. The Basic LINKING protocol has higher computation costs than the GENERATION protocol (see Section 5.5). However, that can be acceptable, because the linking of a pseudonym is a rare case.

Anyway, there is a possibility of optimizing the Basic LINKING protocol. The optimization uses the fact, that all members of a subgroup rooted at node $\langle l, v \rangle$ can compute the same secret shares $s_{\langle l,v \rangle i}$. Therefore, the computation of n hidden secret shares $r_{\langle l,v \rangle i}$ in the first step of the Basic LINKING protocol can be done in parallel by several members of the same subgroup. The possible computation gain is linear proportional to the number of involved members of each subgroup, that we denote as t. This number depends on the chosen level l and height h of $\mathcal{T}$, and is calculated as

$$t \leq 2^{h-l}$$

Therefore, the possible maximal computation gain varies between 1, for $l = h$, and n, for $l = 0$. This optimization can be applied upon the redefinition of the set $\mathcal{K}$ from the previous section, thus

$$\mathcal{K} = \{\mathcal{M}_{K_{\langle l,0 \rangle}} \cup \ldots \cup \mathcal{M}_{K_{\langle l,2^l-1 \rangle}} \mid \mathcal{M}_{K_{\langle l,v \rangle}} \subseteq \mathcal{M} \text{ is a set of } t \text{ members, who all belong to a subgroup rooted at node } \langle l, v \rangle\}$$

The number of members, that participate in the Optimized LINKING protocol is $|\mathcal{K}| = k{\cdot}t$, where k is the number of participants in the Basic LINKING protocol. Each of these members sends her own hidden secret shares during the linking process. This means, that during the optimized protocol every member of $\mathcal{M}$ receives t times more messages, compared to the basic protocol. However, the total length of these messages remains the same, because overall n hidden secret shares must be sent in both protocols. Fig. 9 shows the Optimized LINKING protocol. The computation costs are optimized by factor t.

1. Every member $M_{K_{\langle l,v\rangle}} \in \mathcal{M}_{K_{\langle l,v\rangle}}$
   - computes $\frac{n}{t}$ hidden secret shares $r_{\langle l,v\rangle i}$, using ps
   - combines all $r_{\langle l,v\rangle i}$ to a single message and builds its keyed-hashing value using $K_{\langle 0,0\rangle}$
   - broadcasts the message and its keyed-hashing value with attached real identity
2. Every member $M_s \in \mathcal{M}$
   - verifies keyed-hashing values of received messages and extracts values $r_{\langle l,v\rangle i}$
   - computes product R_i for every other member $M_i \in \mathcal{M}$, $i \neq s$
   - finds public share $S_i = R_i$ and determines member M_i as owner of ps

Fig. 9. Optimized LINKING Protocol

5.4 On Security of the Pseudonym Generation Scheme

In this section we consider different attacks on the Pseudonym Generation Scheme and show its resistance against them. In order to link pseudonym ps_i to the real identity of its holder, adversary must be able to recompute the public share S_i or to compute the corresponding product R_i. Recomputing S_i requires knowledge of all secret shares $s_{\langle l,v\rangle i}$. The security of each secret share is however based on the *Computational Diffie-Hellman Problem* (CDH), since only members, who know the corresponding private key $K_{\langle l,v\rangle}$ or K_i can recompute $s_{\langle l,v\rangle i}$, that is except for M_i only $M_{K_{\langle l,v\rangle}}$ can recompute $s_{\langle l,v\rangle i}$. Thus, passive adversary is not able to compute all secret shares unless the CDH problem can be solved in polynomial time. To compute product R_i, one must know all hidden secret shares $r_{\langle l,v\rangle i}$. Hidden secret shares can only be computed through exponentiation of g with the corresponding $s_{\langle l,v\rangle i}$, that is protected as stated above. Therefore, only shareholders can compute secret shares, thus interaction between all shareholders is required, to be able to link ps_i to its owner M_i.

A possible security risk is given, when adversary M_a knows a secret share $s_{\langle l,v\rangle i}$ between $M_{K_{\langle l,v\rangle}}$ and M_i for some pseudonym ps. M_a can compute the inverse value ps^{-1} and the shared key $skey_{\langle l,v\rangle i} = s_{\langle l,v\rangle i}^{ps^{-1}}$. Using this shared key, M_a can compute secret share $s_{\langle l,v\rangle i}$ for any other pseudonym ps'. As already mentioned, the security of single secret shares is given by the hardness of the CDH problem.

An adversary that aims to forge a pseudonym of the member requires all her shared keys in order to compute the correct public share. As stated above any shared key can only be computed by members that share it. Therefore, the only way for the adversary to forge a pseudonym of the member is to collude with k other members that share keys with the victim.

During the GENERATION protocol every member M_i computes and broadcasts her public share S_i. In the next round she broadcasts her pseudonym ps_i anonymously. Since authentication of broadcasted pseudonyms is not provided, any group member adversary can broadcast a false value for S_i or ps_i. In the linking process none of calculated products R_j would match such S_i. Thus it would be not possible to find out which member has broadcasted the false public share or pseudonym, unless all members reveal their pseudonyms and prove the correctness of their signed shares. Any member, who fails to provide this proof, is an adversary and should be excluded from the communication. In this case every member decides on her own whether to reveal her pseudonym to the others and get free of accusations or to continue hiding her pseudonym and not allow other members to identify her previously broadcasted messages. Therefore, the proposed scheme is applicable in scenarios where members are interested in the linking of their pseudonyms at the end of communication, e. g. auctions or polls.

This weakness of the scheme can be repaired if every member M_s would prove the knowledge of a corresponding link ($ps_s \leftrightarrow S_s$) without revealing it to other group members. In Appendix A we describe a protocol that can be used to prove the knowledge of such link.

At last we discuss some risks given by the collusion of some group members. If all group members that know the secret key $K_{\langle l,v\rangle}$ collude then they can bar the group from the linking process for a pseudonym ps_i by broadcasting the same false value for the hidden secret share $r_{\langle l,v\rangle i}$. In this case other group members are not able to compute the correct value of the product R_i, thus link ps_i to S_i. Our scheme allows optimization against the collusion risk by choosing a lower level l. It is trivial that the lower l is the more group members know the secret key $K_{\langle l,v\rangle}$, thus more group members have to collude.

5.5 On Complexity of the Pseudonym Generation Scheme

In this section we discuss computation and communication costs of our scheme without considering the complexity of the TGDH suite protocols. Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of members participating in the TGDH key agreement protocol, $\mathcal{T}$ the associated IDH key tree with height h, and k, $k = 2^l$ and $0 \leq l \leq h$, a number of required interacting members in the linking process of a pseudonym. The complexity of our scheme for each member is summarized in Table 4.

Table 4. Computation and Communication Costs of the Pseudonym Generation Scheme

	Computation	Communication	
	Exponentiations	Rounds	Messages
GENERATION	$k+1$	2	2
Basic LINKING	$2n$	2	1
Optimized LINKING	$\frac{2n}{t}$	2	1

GENERATION is performed every time a join event occurs or pseudonyms have to be changed after the linking process. In order to save computational power k shared keys $skey_{s\langle l,v\rangle}$ should be computed and stored by M_s after any dynamic event that

causes GENERATION protocol to start. Each member has to perform k group exponentiations to compute all secret shares, and one more to compute her public share. Public shares and pseudonyms should be sent in two separate messages, because the latter requires sender anonymity. Table 4 shows costs of both LINKING protocols. The basic protocol from Fig. 8 requires $2n$ exponentiations, done by every of k participating members. Every member $M_{K_{\langle l,v\rangle}} \in \mathcal{K}$ first computes n hidden secret shares $r_{\langle l,v\rangle i}$, $1 \leq i \leq n$, that require two exponentiations each, and then sends them combined in one message. Every $M_j \in \mathcal{M}$ receives these k messages from members of $\mathcal{K}$ and computes the product R_i. The optimized protocol from Fig. 9 requires $\frac{2n}{t}$ exponentiations, where $1 \leq t \leq n$. Every of t members in $\mathcal{M}_{K_{\langle l,v\rangle}} \subseteq \mathcal{K}$ computes only $\frac{n}{t}$ hidden secret shares and sends them combined in one message. Every $M_j \in \mathcal{M}$ receives these $k \cdot t$ messages from members of $\mathcal{K}$ and computes the product R_i.

The computation costs of the GENERATION protocol vary upon choosing appropriate values for k, according to power limitations of the mobile devices. Because $\mathcal{T}$ is a binary tree, k may only be chosen as a power of two.

6 Conclusion

This paper contains two different contributions concerning the usage of the computation and communication efficient iterative Diffie-Hellman (IDH) protocol for dynamic group key agreement in mobile ad-hoc networks that we have shown on the example of the Tree-based Group Diffie-Hellman (TGDH) protocol suite [7]. The first main result is the proposed communication efficient pseudonym generation scheme that allows members to generate pseudonyms without trusting in any third parties. This scheme can be used in different mobile scenarios, like in director board meetings or spontaneous auctions. Another major contribution is the proposed method of optimization by choosing an appropriate level of the IDH key tree with respect to power limitations of involved mobile devices. This method can be used separately from the pseudonym generation scheme for optimization in other security schemes that are based on threshold revocation.

References

1. N. Bansal and Z. Liu. Capacity, delay and mobility in wireless ad-hoc networks. In *Infocom 2003 (IEEE)*, 2003.
2. C. Becker and U. Wille. Communication complexity of group key distribution. In *ACM Conference on Computer and Communications Security*. ACM Press, NY, November 1998.
3. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology (EUROCRYPT '94),Lecture Notes in Computer Science*, volume 950, pages 275–286. Springer-Verlag Berlin, May 1994.
4. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science*, volume 1294, pages 410–424. Springer-Verlag, 1997.

5. Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *ACM Conference on Computer and Communications Security '00*, pages 235–244. ACM Press, NY, 2000.
6. Y. Kim, A. Perrig, and G. Tsudik. Communication-efficient group key agreement. In *Information Systems Security, Proc. of the 17th International Information Security Conference, IFIP SEC'01*, 2001.
7. Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, 2004.
8. M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8), 2000.

A Proving Public Shares for Pseudonym Generation Scheme

After GENERATION protocol is completed every member M_s knows the set of signed public shares $\mathcal{S} = \{S_i | \ 1 \leq i \leq n\}$ and the set of broadcasted pseudonyms $\mathcal{PS} = \{ps_i | \ 1 \leq i \leq n\}$. Only M_s knows the corresponding link ($ps_s \leftrightarrow S_s$). In order to provide the authenticity of broadcasted pseudonyms, for every $ps_i \in \mathcal{PS}$ it has to be shown that it was used to compute one of known public shares $S_i \in \mathcal{S}$ without revealing that share.

Protocol in Fig. 10 is a proof for this relation. The idea behind the proof is that prover P makes all known public shares $S_i \in \mathcal{S}$ indistinguishable by building corresponding blinded public shares $S'_i = (S_i)^{r^{ps_p}}$ using randomly chosen value r and broadcasting them as a random permutation.

During one run of the protocol P proves depending on the challenge of the verifier V with the probability of $\frac{1}{2}$ either that all S'_i were built as described above using all known S_i or that she knows how a certain pseudonym ps_p relates to one of the broadcasted S'_i. The role of the prover should be taken by a member M_s, that knows the relation between her pseudonym ps_s and public share S_s. The role of the verifier V should be taken separately by all existing group members except for M_s. All verifiers have to agree on the same challenge that P has to respond to, or P has to prove her knowledge to every verifier in a separate protocol run. All messages that P broadcasts are signed using her pseudonym ps_s. Function $\pi : \ \mathbb{N} \rightarrow \mathbb{N}$ specifies a random permutation. The protocol uses zero-knowledge proof techniques $SKROOTLOG$ and $SKLOGLOG$ described by Camenisch *et. al.* in [4]. $SKROOTLOG[\alpha : \ y = g^{\alpha^e}]$ proves the knowledge of an e-th root of the discrete logarithm of y to the base g, where e, g and y are commonly known values. $SKLOGLOG[\alpha : \ y = g^{(a^\alpha)}]$ proves the knowledge of a double discrete logarithm of y to the bases g and a, where a, g and y are commonly known values.

Conjecture 1. *Protocol in Figure 10 fulfils requirements on completeness and soundness.*

Sketch of Proof. Completeness: We consider that both P and V are honest. The completeness of the protocol is obvious for case $c = 0$, since r^{ps_s} was used in step 1 as exponent to compute blinded public shares S'_i in the broadcasted permutation. Therefore, in step 4 the verifier computes the correct values. In order to show the correctness for the case $c = 1$ we examine first zero-knowledge proofs given by values $\mathcal{Z}_1$ and $\mathcal{Z}_2$.

1. P chooses random $r \in_{\mathcal{R}} \mathcal{G}$, computes
$$\{S'_1, \ldots, S'_n\} = \{S_1^{r^{ps_s}}, \ldots, S_n^{r^{ps_s}}\}$$
and broadcasts $\mathcal{S}' = \{S'_{\pi(1)}, \ldots, S'_{\pi(n)}\}$.
2. V broadcasts chosen challenge $c \in_{\mathcal{R}} \{0, 1\}$.
3. Case $c = 0$: P broadcasts
$$\mathcal{X}_0 = r$$
Case $c = 1$:
   - P computes
$$\mathcal{X}_1 = g^{r^{ps_s}},\ \mathcal{X}_2 = \{x_{2_v} |\, x_{2_v} = \mathcal{X}_1^{\left(g^{K_{\langle l,v\rangle}{}^{ps_s}}\right)^{K_s}}, 0 \le v \le 2^l - 1\}$$
$$\mathcal{Z}_1 = SKROOTLOG[r : \mathcal{X}_1 = g^{r^{ps_s}}]$$
$$\mathcal{Z}_2 = \{z_{2_v} |\, z_{2_v} = SKLOGLOG[K_s : x_{2_v} = \mathcal{X}_1^{\left(g^{K_{\langle l,v\rangle}{}^{ps_s}}\right)^{K_s}}], \forall x_{2_v} \in \mathcal{X}_2\}$$
   - broadcasts $(\mathcal{X}_1, \mathcal{X}_2, \mathcal{Z}_1, \mathcal{Z}_2)$
4. Case $c = 0$: V checks
$$\{S_1^{\mathcal{X}_0^{ps_s}}, \ldots, S_n^{\mathcal{X}_0^{ps_s}}\} \stackrel{?}{=} \mathcal{S}'$$
Case $c = 1$:
   - V checks the correctness of $\mathcal{Z}_1$ and computes
$$\mathcal{X}_3 = \{x_{3_v} |\, x_{3_v} = g^{K_{\langle l,v\rangle}{}^{ps_s}}, 0 \le v \le 2^l - 1\}$$
   - checks the correctness of every $z_{2_v} \in \mathcal{Z}_2$
   - checks
$$\prod_{x_{2_v} \in \mathcal{X}_2} x_{2_v} \stackrel{?}{\in} \mathcal{S}'$$

Fig. 10. Proof of S_i in Pseudonym Generation Scheme

The correctness of $\mathcal{Z}_1$ shows that ps_s is the root of the discrete logarithm of $\mathcal{X}_1$ to the base g. Set $\mathcal{Z}_2$ consists of hidden secret shares $r_{\langle l,v\rangle s}$ that are blinded with the random value r^{ps_s}. In order to prove the correctness of each $z_{2_v} \in \mathcal{Z}_2$ verifier has to compute set $\mathcal{X}_3$ using known public keys of all nodes at level l. The correctness of $\mathcal{Z}_2$ shows that every public key $g^{K_{\langle l,v\rangle}}$ at level l was used in the computation of blinded hidden secret shares. At last verifier builds a product of all values $x_{2_v} \in \mathcal{X}_2$. This product corresponds to the blinded public share S'_s of the prover as shown in the following identity:

$$\begin{aligned}
\prod_{x_{2_v} \in \mathcal{X}_2} x_{2_v} &= \prod_v \mathcal{X}_1^{\left(g^{K_{\langle l,v\rangle}{}^{ps_s}}\right)^{K_s}} \\
&= \prod_v \mathcal{X}_1^{g^{K_{\langle l,v\rangle}{}^{K_s ps_s}}} \\
&= \prod_v g^{r^{ps_s} g^{K_{\langle l,v\rangle}{}^{K_s ps_s}}} \\
&= g^{\sum_v \left(g^{r^{ps_s}} g^{K_{\langle l,v\rangle}{}^{K_s ps_s}}\right)}
\end{aligned}$$

$$= (g^{\sum_v (g^{K_{\langle l,v \rangle} K_s ps_s})})^{r^{ps_s}}$$
$$= (S_s)^{r^{ps_s}}$$
$$= S'_s$$

Soundness: We consider information-theoretical soundness of the protocol and show that a cheating prover P^* cannot convince a correct verifier V with more than an exponentially small probability. Consider P^* broadcasting a set of blinded public shares $\{S'_1, \ldots, S'_n\}$ for some chosen pseudonym ps_s. In order to respond to challenge $c = 1$ correctly P^* should be able to compute the sets $\mathcal{X}_2$ and $\mathcal{Z}_2$. This can be done if P^* knows the private key K_s used in the computation of secret shares for the corresponding public share S_s. Since P^* does not have this knowledge he can convince V only on challenge $c = 0$, that is with probability at most $\frac{1}{2}$. Hence, the probability with that P^* convinces V after σ iterations is at most $2^{-\sigma}$. □

The protocol in Figure 10 does not have a zero-knowledge property since the simulator guessing case $c = 1$ should output set ${\mathcal{S}'}_1^{SIM}$ that must be computationally indistinguishable to the set ${\mathcal{S}'}_0^{SIM}$ in case $c = 0$. This is obviously impossible unless the simulator can recompute at least one public share S_i and provide correct values for $\mathcal{X}_2$ and $\mathcal{Z}_2$.