Burmester-Desmedt Tree-Based Key Transport Revisited: Provable Security without Broadcast

Jens-Matthias Bohli[1], María Isabel González Vasco[2], and Rainer Steinwandt[3]

[1] Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe, 76128 Karlsruhe, Germany;
bohli@ira.uka.de

[2] Área de Matemática Aplicada, Universidad Rey Juan Carlos, c/ Tulipán, s/n, 28933 Madrid, Spain;
migonzalez@escet.urjc.es

[3] Dept. of Mathematical Sciences, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA;
rsteinwa@fau.edu

Abstract. A tree-based key transport protocol is presented which can be seen as a generalizing variant of the star- and tree-based protocols proposed by Burmester and Desmedt at EUROCRYPT '94. Our scheme does not rely on the availability of globally verifiable signatures or arbitrary point-to-point connections, and its security against active adversaries is proven in the standard model under the Decision Diffie Hellman assumption.

Keywords: Cryptography, Group Key Establishment, Provable Security

1 Introduction

Group key establishment protocols allow $n \geq 2$ principals to agree upon a common secret key (referred to as the *session key*) for private communication. Differing from group key *agreement* protocols, group key *transport* protocols are key establishment protocols in which a single principal fixes the session key, which is thereafter sent to all the rest. This distinguished participant is usually referred to as the *leader*.

At EUROCRYPT '94, Burmester and Desmedt [BD95] proposed two key transport protocols based on a star and a tree configuration, respectively. In the star-based scheme, the leader has direct communication with all other principals, whereas the latter cannot communicate directly amongst themselves. In the tree-based scheme, the principals' communication channels are represented by a binary tree, rooted by the leader. Thus, in both cases the protocol can be described through a graph, where principals are viewed as nodes and edges represent direct communication channels. Also, both protocols consist of two basic steps: initially, each principal generates his ephemeral secret key-public key Diffie Hellman pair, which he uses for carrying out a Diffie Hellman key exchange with each of his adjacent principals. In a second step, the session key is generated by the leader and transported through the network to each principal. In this transport, the session key is hidden at each edge via the agreed ephemeral keys of the corresponding incident nodes.

Unlike many more recent proposals for key establishment [KY03,KLL04], the mentioned two protocols do not rely on arbitrary point-to-point connections among the principals. Unfortunately, in [BD95] security proofs are not included, and so far these proposals have not been put into a modern framework for analyzing key establishment schemes, say along the lines of [BPR00,BCPQ01,KY03]. As having available only limited network connections imposes additional limitations and difficulties on a key establishment scheme, e. g., in defining session identifiers, a lack of such a formal treatment is rather unfortunate.

In this contribution we introduce a generalization of the mentioned two protocols and give a security proof in the standard model based on the Decision Diffie Hellman (DDH) assumption. Here 'generalization' means that we consider the situation in which the communication graph can actually be any tree, e. g. a spanning tree in an arbitrary connected network. Also, we do not make use of a distinguished principal acting as leader. Our protocol can not be seen as a *centralized group key distribution scheme* (see [KPT04] for an updated definition), as in our proposal any of the principals can start a key transport among an authorized set of connected principals and generate the corresponding key. This somewhat more flexible point of view allows, for instance, to handle hierarchically organized networks, where keys may either be established within a subnetwork (e. g., geographically or organizationally defined) or across several hierarchies.

The main motivation for our construction is that indeed group communication applications often aim at transmitting data using minimum resources. Given a set of n principals, at least $n-1$ direct connections are required for constructing a communication network among all of them. Such a minimal network is thus a tree (i. e., an acyclic connected graph), and the star and tree configurations considered in [BD95] can be taken for special cases.

We consider static groups only, that is, we do not deal with the issue of updating agreed keys when principals leave or join. Also, we do not explore the effect of corrupted insiders[4] or crashed principals (cf. [CS04]). However, we decided to explicitly take into account the following issues which are often taken for granted in proposals for key establishment schemes:

1. It is not assumed that principals know from some protocol-external application already that they want to agree on a key. Instead, the party initiating the key establishment has to inform the other involved parties within the protocol.
2. Besides agreeing on a common key, also a common (session) identifier for this key is to be constructed by the protocol so that applications can later refer to the established session key.

[4] In fact, in our setting where neither global authentication nor a fully connected network is assumed, corrupted insiders are quite powerful

As the security goals of [SSN98] suggest, our protocol will include key confirmation. Indeed, for a group key establishment protocol key confirmation seems to be more desirable than in the two participant case. While for two principals a failure in the key establishment resembles a network failure in the application protocol, in a group key establishment a situation can arise where only a subset of the intended principals share a common key what can be a threat at the application level.

Our protocol requires $\mathcal{O}(n)$ rounds of communication but for $n \gg 2$ needs significantly less messages than constant round protocols like [KY03]. Under the Decision Diffie Hellman assumption it achieves both provable (semantic) security and perfect forward secrecy. Also in the case of active adversaries we decided to do without signatures, as in some applications public verification keys may not be globally available. Instead, we rely on the availability of authenticated links between neighboring nodes in the tree, which are implemented by shared secret keys between neighbors exclusively. We base our proof on a model close to [BCPQ01]. For more clarity we introduce a different notion of correctness and can thereby simplify the definition of partnering.

To avoid ambiguities, in the next section we recall some details of the underlying security model. Thereafter, in Section 3 a basic version of our group key establishment protocol is explained and proven secure against passive adversaries. Finally, Section 4 discusses the active case.

2 Underlying Security Model

For exploring the security of the tree based key transport protocol discussed in the sequel, we essentially follow the approach of Bresson et al. [BCPQ01] and of Katz and Yung [KY03], both building on [BR93,BR95,BPR00]. Although there are clear limitations of this approach (e. g., no malicious insiders are considered), we think it allows a convenient formalization of important security aspects. In [BPR00] session identifiers are introduced for defining partnering, though in many subsequent versions of the model (cf. [KY03,BCPQ01]) slight variants of the definition of partnering are given. Obviously, restrictive definitions of partnering allow for more attacks. On the other hand when using a lenient definition one should make the point that several attacks are excluded, as all oracles that know the session key may be partnered. The latter may severely limit the practical relevance of a security proof. We decided to give a very simple definition of partnering, which together with our definitions of correctness and freshness yields a convenient basis for security proofs. The main ingredients of our security model are as follows.

Participants We consider a fixed set of (potential) protocol participants, $\mathcal{P} = \{U_1, \dots, U_n\}$, which are modeled as probabilistic polynomial time (ppt) interactive Turing machines. Further on, we assume the participants in $\mathcal{P}$ to be connected through a tree-shaped communication network where all communication channels can be used in both directions. In other words, $\mathcal{P}$ is the set of vertices of an undirected tree, whose edges represent the available communication channels.

Each protocol participant U can execute polynomially many protocol instances, usually referred to as *oracles* Π_U^i $(i = 1, 2, \dots)$ in parallel, and messages exchanged throughout the protocol must always be addressed to a specific instance. Intuitively, the oracles Π_U^i can be taken for processes executed by U. Every oracle Π_U^i has assigned the variables state_U^i, sid_U^i, pid_U^i, sk_U^i, term_U^i, used_U^i and acc_U^i:

used_U^i indicates whether this oracle is or has been used for a protocol run;
state_U^i keeps the state information during the protocol execution;
term_U^i shows if the execution terminated;
sid_U^i denotes the unique session identifier, and will act as a name for the key;
pid_U^i stores the set of identities of those principals that Π_U^i aims at establishing a key with—including U himself;
acc_U^i indicates if the protocol instance was successful, i. e. the principal accepted the session key;
sk_U^i stores the session key once it is accepted by the oracle Π_U^i. Before acceptance, it stores a distinguished NULL value.

For more details on the usage of the variables see [BPR00]. As no malicious insiders are considered, we assume an oracle Π_U^i must accept the session key constructed at the end of the corresponding protocol instance if no deviation from the protocol specification takes place. Thus, clearly each sid_U^i must uniquely determine a session key sk_U^i and only principals in pid_U^i may have accepted (at most) one key with the session identifier sid_U^i.

Partnering Two oracles Π_U^i and $\Pi_{U'}^j$ are *partnered* if they have both accepted ($\mathsf{acc}_U^i = \mathsf{acc}_{U'}^j = 1$) and hold the same session identifier $\mathsf{sid}_U^i = \mathsf{sid}_{U'}^j$.

Communication network We assume the communication network to be publicly known and the communication to be organized in rounds. One round consists of the messages that can be sent in parallel and within each round the exact order of message delivery can be fixed arbitrarily. To avoid technical problems like one Turing machine potentially being able to measure the execution time of other parties, we assume that a suitable scheduling mechanism ensures that only one Turing machine is active at a time.

Initialization In case of an active adversary, before the actual key transport protocol is executed for the first time, an initialization phase takes place. Here each neighboring pair of principals $(U, V) \in \mathcal{P}^2$ $(U \neq V)$ is equipped with a uniformly at random chosen common secret key $SK_{\{U,V\}}$ allowing to implement authenticated communication between U and V. An active adversary is not able to influence this initialization phase.

A key motivation to use symmetric keys for implementing authentication are use cases where the available infrastructure is limited and no PKI is available. Even if we cannot assume that principals have access to signature verification keys of all other principals, it may be feasible to implement 'local authentication' with symmetric keys.

Adversarial model When dealing with an active adversary, we may assume she has full control over the communication network, i. e., she can eavesdrop, delay, suppress, and send messages at will. A passive adversary does not interfere with the communication among the parties and thus in her presence all sent messages are delivered as specified in the protocol.

To make the adversary's capabilities explicit, we assume she can access the following *oracles* with the Send and $\mathsf{Corrupt}$ oracles being special in the sense that they are available to an active adversary only:

$\mathsf{Execute}(U_{u_1}, (U_{u_2}, \ldots, U_{u_r}))$ This executes the protocol among unused instances $\Pi_{U_{u_j}}^{i_j}$ of the specified parties and returns a transcript of the protocol run (listing all messages sent during the protocol execution among the oracles $\Pi_{U_{u_j}}^{i_j}$). At this, the principal U_{u_1} initiates the protocol.

$\mathsf{Send}(U, i, M)$ This sends the message M to the instance Π_U^i and outputs the reply generated by this instance. If the adversary calls this oracle with an unused instance Π_U^i and $M = \{U_{u_2}, \ldots, U_{u_r}\}$, then Π_U^i initiates a protocol instance with the partners listed in M. Thereafter, Send returns the 'init message' Π_U^i sends for this purpose.

$\mathsf{Reveal}(U, i)$ yields the session key sk_U^i and the session identifier sid_U^i.

$\mathsf{Corrupt}(U)$ reveals all long term term secret keys $SK_{\{U,*\}}$ of U to the adversary.

$\mathsf{Test}(U, i)$ Only one query of this form is allowed for an active adversary $\mathcal{A}$. Provided that sk_U^i is defined, (i. e. $\mathsf{sk}_U^i \neq$ NULL), $\mathcal{A}$ can execute this oracle query at any time when being activated. Then with probability 1/2 the session key sk_U^i and with probability 1/2 a uniformly chosen random session key is returned.

Correctness To exclude 'useless' protocols, we require that a single execution of the protocol for establishing a key among $U_{u_1}, \ldots, U_{u_r}$ involving the oracles $\Pi_{U_{u_1}}^{i_{u_1}}, \ldots, \Pi_{U_{u_r}}^{i_{u_r}}$ ensures that for the participating oracles that have accepted, i. e., $\{\Pi_{U_{a_1}}^{i_{a_1}}, \ldots, \Pi_{U_{a_s}}^{i_{a_s}}\} = \{\Pi_{U_{u_j}}^{i_{u_j}} \in \{\Pi_{U_{u_1}}^{i_{u_1}}, \ldots, \Pi_{U_{u_r}}^{i_{u_r}}\} | \mathsf{acc}_{U_{u_j}}^{i_{u_j}} = true\}$, the following holds. With overwhelming probability they

- obtain a common session identifier (i.e. $\mathsf{sid}_{U_{a_1}}^{i_{a_1}} = \cdots = \mathsf{sid}_{U_{a_r}}^{i_{a_r}}$) which is globally unique.
- have accepted the same session key $\mathsf{sk}_{U_{a_1}}^{i_{a_1}} = \cdots = \mathsf{sk}_{U_{a_r}}^{i_{a_r}} \neq$NULL associated with the common session identifier $\mathsf{sid}_{U_{a_1}}^{i_{a_1}}$.
- know their partners $\mathsf{pid}_{U_{a_1}}^{i_{a_1}} = \cdots = \mathsf{pid}_{U_{a_r}}^{i_{a_r}} \neq$NULL associated with the common session identifier $\mathsf{sid}_{U_{a_1}}^{i_{a_1}}$ and it is $U_{a_1}, \ldots, U_{a_r} \in \mathsf{pid}_{U_{a_1}}^{i_{a_1}}$.

Remark 1. In case of a passive adversary all of the oracles $\Pi_{U_{u_1}}^{i_{u_1}}, \ldots, \Pi_{U_{u_r}}^{i_{u_r}}$ accept. For an active adversary this cannot be guaranteed, as in this case messages need not to be delivered.

Freshness An instance Π_U^i participating in a key establishment among principals $\mathsf{pid}_U^i = \{U_{u_1}, \ldots, U_{u_r}\} \ni U$ is referred to as *fresh* if none of the following is true:

- For some $U' \in \{U_{u_1}, \ldots, U_{u_r}\}$ a $\mathsf{Corrupt}(U')$ query was executed before a query of the form $\mathsf{Send}(U'', i, *)$ has taken place, where $U'' \in \{U_{u_1}, \ldots, U_{u_r}\}$.
- The adversary somewhen has queried $\mathsf{Reveal}(U, i)$ or $\mathsf{Reveal}(U', j)$ with Π_U^i and $\Pi_{U'}^j$ being partnered.

Remark 2. The first condition in our freshness definition may look overnecessarily restrictive. Note however, that with the more lenient approach proposed in [KY03], an attack of the following type is not excluded: Once the first principal U has computed the session key, U is corrupted and outgoing messages of this party are modified such that the other protocol participants end up with a different session identifier but identical session key. Having provoked the latter situation, breaking the security of the protocol is straightforward.

Security Let us start by fixing some notation: In the sequel, $a \leftarrow A$ denotes either that element a is chosen uniformly at random from A (if A is a set), or random choice of a according to A (if A is a probability distribution). Now, we define—as a function of the security parameter k—the advantage $\mathsf{Adv}_{\mathcal{A}}$ of an adversary $\mathcal{A}$ in attacking a key establishment protocol P (see, again, [KY03]) as

$$\mathsf{Adv}_{\mathcal{A}} := |2\mathsf{Succ} - 1|$$

where

$$\begin{aligned}\mathsf{Succ} := \Pr[&(T, sk_0) \leftarrow \mathsf{P}; sk_1 \leftarrow G; b \leftarrow \{0,1\} : \\ &\mathcal{A}(T, sk_b) = b)].\end{aligned}$$

At this, $G = G(k)$ is the key space and T is the transcript of the protocol run (that is, all the information flowing through the network, which $\mathcal{A}$ can access). Thus, Succ is actually the probability of success $\mathcal{A}$ has on guessing the output of the Test oracle queried on (U, i) such that Π_U^i is fresh.

We say that a protocol P is *secure* provided that, for any ppt adversary $\mathcal{A}$ the function $\mathsf{Adv}_{\mathcal{A}} = \mathsf{Adv}_{\mathcal{A}}(k)$ is negligible. The intuition behind this definition of security is that P is secure against $\mathcal{A}$ if $\mathcal{A}$ cannot distinguish a session key established through P from an element chosen uniformly at random among all possible keys.

Authentication We say that a protocol P achieves *implicit authentication* if once an oracle Π_U^i has accepted a session key, U_i can be sure that no principal outside the intended group has knowledge of this key. *Explicit authentication* is achieved if, in addition, U_i knows that the intended group does have the agreed shared key (i.e., explicit authentication is achieved through implicit authentication and key confirmation).

(Perfect) Forward Secrecy A protocol P is said to achieve *(perfect) forward secrecy* if compromising the long term keys created in the initialization phase does not endanger previously established session keys.

3 A Basic Tree Based Key Transport Protocol

We describe here our tree based group key transport protocol, which is inspired in the star and tree protocols of Burmester and Desmedt [BD95].

3.1 Description of the Basic Protocol

Let $\mathcal{P}$ be a fixed finite set of principals, engaged in a connected communication network. The protocol described below can be applied to arbitrary connected subgraphs of $\mathcal{P}$; let $\mathcal{P}'$ be the set of principals (vertices) in such a subgraph and denote the cardinality of $\mathcal{P}'$ by n. We assume that some spanning tree among $\mathcal{P}'$ is globally known. This tree is used by our protocol for exchanging messages among principals in $\mathcal{P}'$. Finally, let us also assume that a cyclic group G of prime order q and a generator g of G have been fixed, so that the corresponding Decision Diffie Hellman problem is hard. Now consider the following protocol P:

Basic Protocol P:

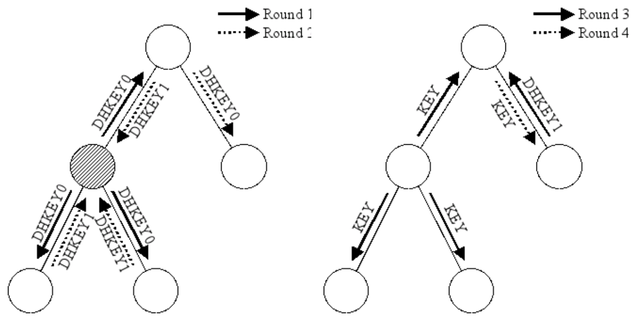
Let us suppose principal U wants to establish a key among a set of principals $\mathcal{P}' \ni U$ forming the vertices of a subtree of the communication network, and take U for the root of this tree. Then U selects uniformly at random a session key $\mathsf{sk} \in G$ and generates at random a session identifier sid (say a bitstring of length linear in the security parameter, whose first bits are an encoding of $U's$ identity and the rest is chosen uniformly at random). Also, he chooses uniformly at random an exponent $a_1 \in \mathbb{Z}_q$, computes $g_1 = g^{a_1}$ and sends $(\mathsf{DHKEY0}, g_1, \mathsf{pid})$ to all his direct descendants.

Each principal U_i, upon receiving this message, chooses $a_i \in \mathbb{Z}_q$ uniformly at random and

- sends $(\mathsf{DHKEY1}, g^{a_i})$ to his parent, thereby establishing a common key between U_i and his parent;
- sends $(\mathsf{DHKEY0}, g^{a_i}, \mathsf{pid})$ to all his direct descendants, thereby initiating a Diffie Hellman key exchange.

Proceeding in this way 'down the tree', after no more than n rounds each principal in $\mathcal{P}'$ has established a common Diffie Hellman key with his parent as well as a common Diffie Hellman key with each of his direct descendants. Once U receives an answer $(\mathsf{DHKEY1}, g^{a_i})$ from a direct child U_i, in the next round he sends $(\mathsf{KEY}, \mathsf{sid}, g^{a_1 a_i} \cdot \mathsf{sk})$ to U_i, from which U_i can extract the session key sk and accept it. The transmission of $(\mathsf{KEY}, \mathsf{sid}, g^{a_1 a_i} \cdot \mathsf{sk})$ to U_i takes place in the same round in which U_i receives the answers $(\mathsf{DHKEY1}, g^{a_{i_j}})$ from his direct descendants. Thus, using the Diffie Hellman keys established with his direct descendants, in the next round each U_i can communicate sid and sk to his direct descendants U_{i_j}. In summary the $(\mathsf{sid}, \mathsf{sk})$-transmissions are always 'one level behind' the Diffie Hellman key establishments in the tree, and after at most $n+1$ rounds of communication every principal has learned $(\mathsf{sid}, \mathsf{sk})$.

A sample protocol run is sketched in Figure 1.



(a) Round 1 and 2 (b) Round 3 and 4

Fig. 1. The basic protocol, started by the hatched node.

3.2 Analysis in the Passive Case

Correctness In the passive case the correctness is straightforward to check, as the adversary cannot interfere with the protocol execution and the protocol clearly distributes sk, sid and pid among the participants.

Security As depicted in Section 2, we essentially follow the standard adversarial model of Bresson et al. [BCPQ01]. For a fixed probability distribution χ on the set of transcript-session key pairs, we denote by $\Pr[(T, \mathsf{sk}) \leftarrow \chi : \mathcal{A}(T, \mathsf{sk}) = 1]$ the probability of success an adversary $\mathcal{A}$ has, in sight of a pair (T, sk) in distinguishing whether $\mathsf{sk} \in G$ is a uniformly at random chosen key or a properly constructed key on a protocol run with transcript T (for instance, if χ is the distribution induced by P, the above probability denotes Succ).

Proposition 1. *Assume that the DDH assumption holds. Then the tree based protocol* P *described above is a secure group key establishment protocol; namely, for any ppt passive adversary* $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}(t)$ *is negligible.*

Proof. Let us start by noting that for the case of passive adversaries it is sufficient to consider only $\mathsf{Execute}$ oracle calls of the adversary. Also, it is sufficient to consider only a single call to the $\mathsf{Execute}$ oracle—as in protocol P no shared information/state among different protocol executions exists, information collected from additional protocol instances can be simulated by the adversary herself. Analogously, we do not have to be concerned for Reveal calls either, because of session keys being chosen uniformly at random in each protocol instance (therewith being statistically independent).

Let $\mathcal{A}$ be a ppt passive adversary achieving an advantage in attacking the protocol (i. e., distinguishing random group elements from the session key). We can transmogrify $\mathcal{A}$ into a DDH adversary $\mathcal{D}$ as follows. $\mathcal{D}$ uses $\mathcal{A}$ as black box and simulates the $\mathsf{Execute}$ oracle as required. Presented a DDH instance (g^A, g^B, g^C), where dependent on a random coin, C is either AB or a random value, $\mathcal{D}$ proceeds as follows. As soon as $\mathcal{A}$ calls the $\mathsf{Execute}$ oracle, $\mathcal{D}$ generates a protocol transcript, where in the transport phase the Diffie Hellman messages g^{a_i} of U_i are replaced by $(g^A)^{a_i}$ if U_i is located at even distance to the principal that initiated the protocol session. For those U_i located at odd distance, $\mathcal{D}$ replaces g^{a_i} with $(g^B)^{a_i}$. Further on, instead of sending the encrypted session key $(\mathsf{KEY}, \mathsf{sid}, g^{Aa_iBa_j} \cdot \mathsf{sk})$ to U_j, $\mathcal{D}$ sends $(\mathsf{KEY}, \mathsf{sid}, (g^C)^{a_ia_j} \cdot \mathsf{sk})$ to U_j. Note that the session transcript obtained in this way is valid, because sk is a uniformly at random chosen group element.

Once $\mathcal{A}$ queries the Test oracle, $\mathcal{D}$ forwards sk to $\mathcal{A}$. If $C = AB$, then $\mathcal{A}$ received the correct session key. In case of C having been chosen uniformly at random, the sk-value appears to $\mathcal{A}$ as a uniformly at random chosen group element. Thus, by adopting the answer of $\mathcal{A}$, the DDH adversary $\mathcal{D}$'s advantage in solving the DDH problem equals $\mathcal{A}$'s advantage in attacking the protocol. Thus, under the DDH assumption a ppt passive adversary $\mathcal{A}$ achieving a non-negligible advantage in attacking P cannot exist.

Authentication We may assume implicit authentication is obtained, as all principals are supposed to behave honestly and the adversary can not impersonate anyone (for she cannot send any message). Explicit authentication is achieved in the sense that we know that indeed all group members will finally get the key, as the adversary is passive and therewith delivery of all messages is guaranteed.

Communication complexity As we are doing without a broadcast channel, in addition to the round complexity $\mathcal{O}(n)$ also the total number of messages needed by the above protocol is of interest. By construction this depends on the number of protocol participants only. Namely, for establishing the Diffie Hellman keys we need $2 \cdot (n-1)$ messages, and for transporting the actual session key, another $n-1$ messages are sent. Thus, in total the passive protocol needs $3n-3$ messages.

4 Allowing Active Adversaries

4.1 A Modified Protocol

For dealing with active adversaries we have to augment the above protocol with the initialization phase described in Section 2. Thus, upon a $\mathsf{Corrupt}(U_i)$ query of the adversary $\mathcal{A}$, all long-lived keys authenticating messages between U_i and its neighboring principals are revealed to $\mathcal{A}$.

Based on this 'infrastructure' we extend the protocol from Section 3 as follows:

Starting – key transport phase:

Let us suppose principal U wants to establish a key among a set of principals $\mathcal{P}' \ni U$ forming the vertices of a subtree of the communication network, and take U for the root of this tree. Then U selects uniformly at random a session key $\mathsf{sk} \in G$ and generates at random a session identifier sid (say a uniformly at random chosen bitstring of length equal to the security parameter k, prepended by an encoding of U's identity). Also, he chooses uniformly at random an exponent $a_1 \in \mathbb{Z}_q$, computes $g_1 = g^{a_1}$ and sends an authenticated packet $(\mathsf{DHKEY0}, \mathsf{sid}, \mathsf{pid}, g_1, U)$ to all his direct descendants.

Each principal U_i, upon receiving this message first checks the validity of the authentication of U and ignores the packet if this check fails. If the packet is properly authenticated, U_i chooses $a_i \in \mathbb{Z}_q$ uniformly at random and

- sends an authenticated packet having the form $(\mathsf{DHKEY1}, \mathsf{sid}, g^{a_i}, U_i)$ to his parent, thereby establishing a common key between U_i and his parent;
- sends an authenticated packet having the form $(\mathsf{DHKEY0}, \mathsf{sid}, \mathsf{pid}, g^{a_i}, U_i)$ to all his direct descendants, thereby initiating a Diffie Hellman key exchange.

Proceeding in this way 'down the tree,' after no more than n rounds each principal in $\mathcal{P}'$ has established a common Diffie Hellman key with his parent as well as a common Diffie Hellman key with each of his direct descendants. Once U receives an answer g^{a_i} from a direct descendant U_i, in the next round he sends an authenticated packet $(\mathsf{KEY}, \mathsf{sid}, g^{a_1 a_i} \cdot \mathsf{sk}, U)$ to U_i, from which U_i can—after having checked the validity of the authentication—extract the session key sk. The transmission of the tuple $(\mathsf{KEY}, \mathsf{sid}, g^{a_1 a_{1i}} \cdot \mathsf{sk}, U)$ to U_i takes place in the same round in which U_i receives the answers $(\mathsf{DHKEY1}, \mathsf{sid}, g^{a_{1i_j}}, U_{i_j})$ from his direct descendants. Thus, using the Diffie Hellman keys established with his direct descendants, in the next round U_i can communicate sid and sk to his direct descendants U_{i_j}. In summary the $(\mathsf{sid}, \mathsf{sk})$-transmissions are always 'one level behind' the Diffie Hellman key establishments in the tree, and after at most $n+1$ rounds of communication every principal has learned sid and sk.

Key confirmation phase: For this part of the protocol, we again imagine the subtree formed by $\mathcal{P}'$ as being rooted in the principal U that initiated the protocol run.

1. Once a protocol participant U' has learned both sid and sk, we distinguish three cases:
   - If U' is a leaf, he sends an authenticated acknowledgment packet $(\mathsf{ACK}, \mathsf{sid}, U')$ to his parent.

- If U' is an inner node, he waits until having received correctly authenticated acknowledgment packets $(\mathsf{ACK}, \mathsf{sid}, U'_i)$ from all his direct descendants. Then he sends an authenticated acknowledgment packet $(\mathsf{ACK}, \mathsf{sid}, U')$ to his parent.
- If U' is the root, he he waits until having received correctly authenticated acknowledgment packets $(\mathsf{ACK}, \mathsf{sid}, U'_i)$ from all his direct descendants. Then he accepts the session key sk and sends a correctly authenticated confirmation packet $(\mathsf{CNF}, \mathsf{sid}, U')$ to all his direct descendants.

2. Once a principal U_{ij} receives a correctly authenticated confirmation packet $(\mathsf{CNF}, \mathsf{sid}, U_i)$ from his parent he accepts the session key sk and sends a correctly authenticated confirmation packet $(\mathsf{CNF}, \mathsf{sid}, U_i)$ to all his direct descendants.

In the worst case (a tree consisting of a single branch of length n) after $2 \cdot (n-1)$ rounds the confirmation messages have been delivered throughout the network.

4.2 Analysis in the Active Case

Correctness Proving the above protocol to be correct is not hard:

Proposition 2. *The above modified protocol for authenticated group key establishment is correct in the presence of active adversaries under the existential unforgeability of the authentication scheme under chosen message attacks and achieves forward secrecy.*

Proof. The session identifier is generated uniformly at random by the initiator so it is globally unique with overwhelming probability. The session identifier and the partner identifier resp. the session identifier and the session key are sent in the same message that cannot be forged by the adversary. So, since all participants are supposed to be honest, they will accept the same session identifier, partner identifier and session key. Finally the session key is only sent to participants that are included in the partner identifier and all participants will know their partners. Clearly, the modified protocol achieves forward secrecy, as the long-term secret keys are used for message authentication only.

Security The security of the above protocol is summarized in

Proposition 3. *The above modified protocol for authenticated group key establishment is secure against active adversaries under the DDH assumption and the existential unforgeability of the authentication scheme under chosen message attacks.*

Proof. The proof comprises a series of games similar to [KLL04]. Let $\mathsf{Adv}_{\mathrm{S}}^{\mathrm{cma}}$ be the maximum advantage that a ppt adversary achieves in forging a message/MAC pair under chosen message attack and let Forge be the event that $\mathcal{A}$ outputs a new authenticated message $(m, MAC_{SK_{\{U_i,U_j\}}}(m))$ with respect to the key $SK_{\{U_i,U_j\}}$ shared between U_i and U_j without having queried $\mathsf{Corrupt}(U_i)$ or $\mathsf{Corrupt}(U_j)$ before. Let Repeat be the event that the principal U starting the protocol chooses a session identifier sid that was previously or is currently used for another protocol run. Since sid is chosen (uniformly) at random by U who is supposed to be uncorrupted we have $P(\mathsf{Repeat}) \leq \frac{(q_s+q_{ex})^2}{2^{k+1}}$ with q_s resp. q_{ex} denoting the number of calls to the Send resp. $\mathsf{Execute}$ oracle.

Now assume the existence of an adversary $\mathcal{A}$ achieving an advantage $\mathsf{Adv}_{\mathcal{A}}$ in attacking the modified protocol. Without restriction we assume $\mathsf{Succ} > 0.5$, thus $\mathsf{Adv}_{\mathcal{A}} = 2 \cdot \mathsf{Succ} - 1$.

- **Game 0:** This game is identical to the real attack with the oracles faithfully simulated for the adversary $\mathcal{A}$. The probability of a successful attack is denoted by Succ_0 and is identical to the success probability of the adversary attacking the real protocol:

$$\mathsf{Succ}_0 = \mathsf{Succ}.$$

- **Game 1:** This game differs from Game 0 if the event Forge occurs. In this case Game 1 will be aborted. An adversary $\mathcal{A}$ that succeeds in Game 0 but not in Game 1 has provoked the event Forge. Thus

$$\mathsf{Succ}_0 - \mathsf{Succ}_1 \leq P(\mathsf{Forge}).$$

  To use adversary $\mathcal{A}$ for forging a message authentication for a given authentication key, this key has to be assigned to one of the n principals. If $\mathcal{A}$ produces a forgery the probability that it is for the selected principal is $\frac{1}{n}$. Thus:

$$\mathsf{Adv}_{\mathrm{S}}^{\mathrm{cma}} \geq \frac{1}{n} \cdot P(\mathsf{Forge}).$$

- **Game 2:** Game 2 is, compared to Game 1, aborted if the event Repeat occurs. Hence an adversary $\mathcal{A}$ successful in Game 1 but failing in Game 2 has provoked the event Repeat.

$$\mathsf{Succ}_1 - \mathsf{Succ}_2 \leq P(\mathsf{Repeat}) \leq \frac{(q_s + q_{ex})^2}{2^{k+1}}.$$

Due to the definition of freshness, the adversary cannot forge a message without violating freshness of the oracles participating in the protocol instance. Moreover, as each protocol instance among fresh oracles has a unique session identifier and the messages within a protocol instance are distinguished the adversary cannot reuse old messages. All fresh oracles that accept the same session key sk hold the same sid because they are sent in the same message. Therewith all of these oracles are partnered and the sk of the test session cannot be revealed to the adversary.

The adversary $\mathcal{A}$ that successfully attacks Game 2 can be used to solve the DDH problem, just as shown in the proof of Proposition 1 (note that the situation she faces is exactly the case of a passive adversary attacking the basic protocol presented in the previous section, except that in this case she can refuse delivery of messages and halt the protocol if it was initiated by a Send call):

$$\mathsf{Succ}_{\mathrm{DDH}} \geq \mathsf{Succ}_2.$$

Putting it all together we obtain:

$$\mathsf{Succ}_0 \leq n \cdot \mathsf{Adv}_{\mathrm{S}}^{\mathrm{cma}} + \frac{(q_s + q_{ex})^2}{2^{k+1}} + \mathsf{Succ}_{\mathrm{DDH}},$$

thus,

$$\mathsf{Adv}_{\mathcal{A}} \leq 2n \cdot \mathsf{Adv}_{\mathrm{S}}^{\mathrm{cma}} + \frac{(q_s + q_{ex})^2}{2^{k}} + \mathsf{Adv}_{\mathrm{DDH}},$$

that is, the advantage of $\mathcal{A}$ in attacking the modified protocol is negligible.

Authentication Implicit authentication is obtained: no principal outside the intended group can learn the key. Moreover, by key confirmation the protocol achieves explicit authentication.

Communication complexity The message flow in the key transport phase is basically identical to the message flow in the passive protocol and takes $3 \cdot (n-1)$ messages. In the key confirmation phase each principal has to send one confirmation message to each of his neighbors. Thus, the key confirmation phase takes another $2 \cdot (n-1)$ messages, yielding a total of $5n-5$ messages for the modified protocol.

5 Conclusion

The above discussion shows that the original tree based protocols of Burmester and Desmedt can be generalized to a group key establishment protocol that is provably secure in the standard model and that works in a rather general setting: neither a completely connected network nor global authentication are necessary. Moreover, the (linear) message complexity of the protocol is rather moderate, thereby facilitating its use in applications with a restricted communication infrastructure.

Acknowledgments

This work has been partially supported by the German Academic Exchange Service DAAD and the Spanish M.E.C. as part of the BaSe CoAT project within the Acciones Integradas Hispano-Alemanas.

References

[BCPQ01] E. Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquartier. Provably Authenticated Group Diffie-Hellman Key Exchange. In P. Samarati, editor, *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS-8)*, pages 255–264. ACM, 2001.

[BD95] M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 1995.

[BM04] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2004.

[BPR00] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT'00*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, 2000.

[BR93] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993.

[BR95] M. Bellare and P. Rogaway. Provably secure session key distribution— the three party case. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing, STOC'95*, pages 57–66. ACM Press, 1995.

[CS04] C. Cachin and R. Stobl. Asynchronous Group Key Exchange with Failures. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 357–366. ACM, 2004.

[KLL04] H. J. Kim, S. M. Lee, and D. H. Lee. Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In P.J. Lee, editor, *Advances in Cryptology — ASIACRYPT'04*, volume 3329 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2004.

[KPT04] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Trans. Inf. Syst. Security*, 7(1):60–96, 2004.

[KY03] J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In D. Boneh, editor, *Advances in Cryptology — CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2003.

[SSN98] S. Saeednia and R. Safavi-Naini. Efficient identity-based conference key distribution protocols. In C. Boyd and E. Dawson, editors, *Information Security and Privacy: Third Australasian Conference — ACISP'98*, volume 1439 of *Lecture Notes in Computer Science*, pages 320–331. Springer, 1998.