A Token Based Key Distribution Protocol for Closed Group Meetings

Fuwen Liu, Hartmut Koenig
Brandenburg University of Technology Cottbus
Department of Computer Science
PF 10 33 44, 03013 Cottbus, Germany
{lfw,koenig}@informatik.tu-cottbus.de

Abstract. Many emerging interactive and collaborative applications use the peer-to-peer paradigm nowadays. In every-day life peer-to-peer meetings of small groups are dominant, e.g. for business talks. Confidentiality is of primary concern in this context to provide group privacy. To assure confidentiality the partners have to agree upon a secret group key for encrypting their communication. This requires a secure distributed group key exchange protocol which assures that only active, uniquely authenticated group members know the current session key. In this paper we present a novel distributed key distribution protocol, called TKD, to efficiently support the key renewal in small dynamic peer groups. Performance comparisons show that TKD has a lower key refreshment delay compared to existing key exchange protocols.

1 Introduction

Nowadays modern group oriented applications tend to apply the peer-to-peer paradigm to be independent of a centralized group server representing a single point of failure. Decentralized managed groups are more flexible. They better support spontaneity and mobility to ad hoc set up meetings at varying locations. Such settings though make new demands on group security. Especially applications such as audio/ video conferences have to provide group privacy and data integrity if they are deployed in business meetings.

In order to assure confidentiality in a meeting the partners have to agree upon a common secret key for encrypting their communication. It is intuitive that a decentralized key management protocol in which members themselves manage the group key should be deployed in peer-to-peer systems. In particular, real-time settings strongly require efficient decentralized key exchange protocols to minimize the interference period in group communication caused by the key refreshment, because in the asynchronous Internet hosts are usually unable to synchronously update their group key [1, 2].

In this paper we present a novel distributed key distribution protocol, called TKD (*token based key distribution*) to efficiently support the key renewal in small dynamic peer groups. We focus on closed dynamic peer groups of less than 100 participants here. The term *closed* indicates in this context that only current group members are allowed to send messages to the group. The entrance into the meeting is by invitation like oral or written invitations in every-day life. Many of these meetings such as business talks, project meetings, consultations, teleseminars, multi-party games, and others have usually only a small number of participants. Larger group meetings are usually managed in a hierarchical way rather than peer-to-peer [26]. However, small group peer-to-peer meetings are the dominant kind of meeting in every-day life. Simple and efficient features are required for their implementation on the Internet.

The paper is organized as follows. After addressing related work in Section 2 we describe the principle of the TKD protocol in Section 3. Next, Section 4 evaluates its performance compared to other key exchange protocols. In Section 5 we sketch how TKD fulfills the security demands. Some final remarks conclude the paper.

2 Related work

Group key management protocols can be generally classified into centralized and distributed protocols [3] depending on the fact whether the group key renewal is solely managed by a single entity (e.g. key server) or collaboratively performed by the group members themselves.

The *Group Key Management Protocol* (GKMP) is the simplest centralized approach used for the group key management [4]. The key server agrees upon a secret key with each group member. It delivers the new group key to each member encrypted with the corresponding secret key whenever required. This scheme is not efficient, because it requires $O(n)$ messages and $O(n)$ encryption cost for a rekeying event. Wong et al. proposed the *Logical Key Hierarchy* (LKH) protocol [5] which reduces the number of rekeying messages and the number of encryption from $O(n)$ in GKMP to $O(\log n)$. In this scheme the key server maintains a tree of keys so that for each group key refreshment the key server needs only to change the keys on the path from an affected leaf to the root. It is worth to mention that the rekeying efficiency of LKH mainly relies on a balanced key tree. After many rekeying operations the key tree may become imbalanced. To keep the efficiency of LKH it is necessary to rebalance the key tree [6].

Distributed group key management protocols can be divided into two groups: group key agreement and group key distribution protocols [7]. *Group key agreement protocols* are based on the original two-party Diffie-Hellman key exchange protocol [8]. Their basic idea is that each group member has to contribute a share to generate the group key. When the group membership changes, a group member is selected to compute new intermediate keys and distribute them to the group. Based on these intermediate keys and its own share each group member can independently compute the group key. Examples of such protocols are BD [9], CLIQUES [10], and TGDH [11]. The latter proved to be the most efficient one among these protocols related to computational and communication overhead [7]. For each group key renewal in TGDH, a defined group member, the so-called *sponsor*, generates the new interme-

diate keys and distributes them to the group over public channels. Each member computes the new group key using these intermediate keys and its own share.

In contrast to group key agreement protocols, *group key distribution protocols* dynamically select a group member to generate and distribute the new group key. Examples of such protocols are DTKM [12] and the proposal of Rodeh et al. [13]. The latter is more efficient than the DTKM protocol, because it only needs two communication rounds to complete the key renewal, while DTKM demands $log_2 n$ rounds. In the Rodeh protocol all used keys are arranged in a key tree. The leaves of the tree correspond to group members. The left-most leaf is defined as the *tree leader*. When renewing the key the tree leader generates the new group key and sends it to the subtree leaders over secure channels. The subtree leaders forward the new group key to their respective subtree members.

Key tree based protocols like LKH have been proven to be an appropriate solution for a centralized group key management, also for small groups [5]. Several distributed protocols like TGDH and the Rodeh protocol borrowed the key tree concept. Is the key tree based protocol an appropriate approach for small groups? In the sequel we propose an alternative approach for small dynamic peer groups and show that it is more efficient than key tree based one.

3 TKD Protocol

In this section we give an overview of the basic principle of the TKD approach. First we introduce the system architecture assumed for TKD.

System architecture. TKD assumes a three-layer architecture which consists of an application layer, a security layer, and a group communication layer (see Figure 1).

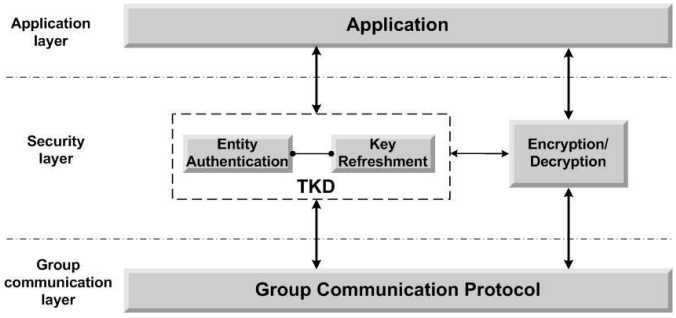

Fig. 1. System architecture

The *application layer* is not further specified here. We assume that it contains the control modules of the given application such as QoS management or floor control modules.

The *security layer* consists of two parts: the encryption/decryption module for data and media exchange, and the TKD protocol which distributes the group key used in the encryption/decryption function and authenticates joining members. The security

layer is closely connected with the group communication layer which assures the consistency of the group data.

The *group communication layer* forms the basis for reliable, collaborative peer-to-peer applications. In our model we assume that the group communication layer indicates all changes in the group composition (join, leave, or failure of peers) to the upper layers to equally keep the membership consistent. Thus all peers have the same view on the actual group state and can uniquely decide all group related issues by themselves, e.g. QoS parameter settings, floor assignment, or key refreshment. To achieve this goal the group communication protocol should provide virtual synchrony [14] which assures that all recipients of the same group membership reliably receive messages in the same order as they are sent. This requires that a group communication protocol should be *reliable* to ensure that no data is lost, *ordered* to ensure that data are delivered in the order as they are sent, and *atomic* to ensure that all peers are updated equally. There are several protocols which meet these requirements like RMP [15], the Totem Protocol [16], Ensemble [17], Spread [18], and GCP [19]. Decentralized key management protocols heavily depend on the virtual synchrony property of the group communication protocol for refreshing the group key [9, 5, 11]. If this property is not provided, members may have a different view on the group membership when a key renewal is required. This may lead to confusions in the group key renewal process, since more than one member may be selected to generate the group or intermediate keys, respectively. Therefore, we assume like other decentralized key management protocols that a group communication protocol with virtual synchrony is applied in the communication layer.

We further assume that the group management, which executes the invitation procedure, is contained in this layer. Furthermore, all participants belong to an identical trust infrastructure or namespace. The group is set up by an initiator which invites the partners. Later further partners can join the group if desired. The decision to invite new partners is based on social agreement of all partners.

Principle of TKD. TKD is a token based protocol. The group members form a logical ring based on the group membership list generated in the group communication layer. The token determines the group member that generates a new group key and initiates the key distribution procedure. The group key is renewed whenever the group composition changes. The token principle was chosen to select the member responsible for the group renewal process in this dynamic group configuration. For smaller groups, as assumed here, the token approach is efficient enough. The token holder is also the group member who authenticates the joining partners. We further assume that an authenticated member in a closed group meeting is trustworthy, i.e. he/she does not actively attempt to disturb the system and to disclose the group key to non-members. No assumptions are made on the trustworthiness of partners after leaving. These assumptions correspond to practical security applications. Other decentralized group key protocols as discussed above rely on similar assumptions. The initiator of the group creates the first token. After renewing the group key the token holder hands the token over to the next group member in the ring. The token shift is part of the rekeying message which is multicast to the whole group. Thus each group member knows the current token holder at any time. The reliable delivering of the rekeying message is guaranteed by the underlying group communication layer as discussed above.

The group key renewal is based on the Diffie-Hellman (DH) key exchange principle [8]. After generating a new group key the token holder establishes temporary secure channels to all members to deliver the key. For this, it uses the shared DH secrets, which are shared with each other member, and a nonce, which is only valid for this group key renewal cycle. The details are given for the join and the leave procedure next.

Join procedure. The join procedure consists of two steps: (1) the authentication phase in which the invitee and the token holder mutually authenticate and (2) the proper join phase causing the group key refreshment (see Figure 2). Five messages or rounds, respectively, are needed for the join procedure: four rounds for authentication and one for the key refreshment.

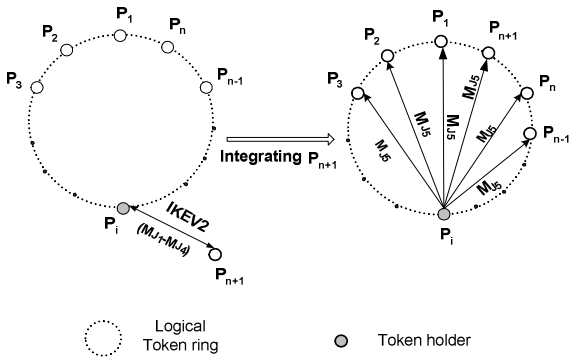

Fig. 2. Join Procedure

Unlike most group key management protocols TKD introduces a **partner authentication**. It is supposed to assure that the group key is only delivered to an authenticated member and that the new member can be sure that the received key is in fact sent by the inviting group. We apply the newly proposed internet draft standard IKEv2 [21] due to its increased efficiency, security, flexibility and robustness compared to the predecessor IKE [20]. Let us assume that P_{n+1} is invited to join a group of n participants: $P_1 \ldots P_n$ (see Figure 2). The token holder is P_i. The group members including the token holder P_i are informed of the acceptance of the invitation by the underlying group communication protocol. Note that the token holder is not necessarily identical with the inviting participant. Thus it is avoided that this participant invites people whom are not agreed upon. To accomplish the mutual authentication between P_i and P_{n+1} the four messages (M_{J1}-M_{J4}) have to be exchanged in accordance with IKEv2.

$$M_{J1}(P_i \rightarrow P_{n+1}) : HDR, KE_i, SA_i, NA_i$$
$$M_{J2}(P_{n+1} \rightarrow P_i) : HDR, KE_{n+1}, SA_{n+1}, NA_{n+1}$$
$$M_{J3}(P_i \rightarrow P_{n+1}) : HDR, SK\{ID_i, CERT_i, SIG_i, ID_1, ID_2 \ldots ID_n, g^{r1}, g^{r2} \ldots g^{rn}\}$$
$$M_{J4}(P_{n+1} \rightarrow P_i) : HDR, SK\{ID_{n+1}, CERT_{n+1}, SIG_{n+1}, g^{rn+1}\}$$

With the exchange of messages M_{J1} and M_{J2}, the token holder and the invitee negotiate the security association *SA* which specifies the cryptography parameters used for the messages M_{J3} and M_{J4}. *Further* they generate the shared keys *SK* and *SK`* on the basis of the exchanged public DH values $KE_i$ and $KE_{n+1}$ as well as a nonce *NA*. The shared keys are used to protect the subsequent messages $M_{J3}$ and $M_{J4.}$ To avoid a reflection attack, separate session key *SK* and *SK`* are used for each direction [21]. *SK* (and also *SK*') consists of the encryption key Sk_e and the authentication key SK_a which are generated according to the IKEv2 draft [21].To adapt to the group communication scenario some message components was added to messages M_{J3} and M_{J4} to exchange information between the token holder and the invitee. The token holder delivers the group information to the new member in message M_{J3} including all members' identities (ID_1, ID_2,…, ID_n) and the respective public DH values (g^{r_1}, g^{r_2},…g^{r_n}). The invitee P_{n+1} returns its identity ID_{n+1} and its public DH value $g^{r_{n+1}}$ with message M_{J4}. Upon receipt of message M_{J3} the invitee P_{n+1} performs *n* DH agreements to get the shared DH secrets with the other members. Virtually the new member does not have to compute these *n* DH agreements in real-time. This can be done "off-line", because the new member will apply the shared DH secrets only when it becomes the token holder to establish the temporal secure channels.

If the token holder fails to authenticate the invitee it notifies the group about this and hands the token over to its neighbor ID_{i+1} with message $M_{Jf,}$:

$$M_{Jf}(P_i \rightarrow P_1, P_2 ... P_n) : HDR, GK_{old}\{Token, VT, ID_{i+1}, Authf, ID_{n+1}\}$$

where *Authf* indicates the failed authentication, ID_{n+1} is the new member's identity. The token version VT is used to prevent replay attacks. It is increased by 1 each time the token is forwarded. When receiving $M_{Jf,}$ each member knows that the new member failed to join the group and who is the new token holder. All members keep the group key unchanged.

After successfully authenticating the invitee the token holder P_i starts the **group key renewal**. The new key GK_{new} is randomly generated and thus independent of previously used keys. The token holder multicasts the new key GK_{new} together with the token in the rekeying message M_{J5} to the new partner P_{n+1} and the other members (see Figure 2). For the latter, temporal secure channels are established as described below. Message M_{J5} has the following format:

$$M_{J5}(P_i \rightarrow P_1, P_2 ... P_{n+1}) : HDR, GK_{old}\{ID_i, N_i\}, K_{i1}\{VK, GK_{new}\}$$
$$,..., K_{in}\{VK, GK_{new}\}, SK\{GK_{new}, VK, GSA, ID_i\}, \mathrm{GK}_{new}\{Token, VT, ID_{i+1}, g^{r_{n+1}}, ID_{n+1},\}$$

The message consists of four parts which serve different purposes. The first part of message M_{J5} contains token holder's identity ID_i and a nonce N_i used to construct the temporal secure channels between the token holder and the other members. This part is encrypted with the old group key GK_{old}. The second part of message M_{J5} contains the new group key GK_{new} and the group key version *VK*. The group key version distinguishes old group keys from the new one and is used to counter replay attacks. *VK* is increased by 1 each time the group key is renewed. These data are separately encrypted for each group member using its temporal channel keys K_{ij} (j=1, 2,…n and $j \neq i$) between the token holder P_i and the other group members. The token holder P_i

creates a temporary secure channel K_{ij} with each group member using the shared secrets with them and the nonce N_i. The secure channels are established as follows:

$$K_{ij-e} = HMAC(g^{r_i r_j}, g^{r_i r_j} N_i\ ID_i\ ID_j\ 0) \quad (1)$$

$$K_{ij-a} = HMAC(g^{r_i r_j}, g^{r_i r_j} | N_i | ID_i | ID_j | 1) \quad (j=1, 2,\ldots.n \text{ and } j \neq i) \quad (2)$$

N_i and ID_i are the nonce and token holder's identity contained in the first part of message M_{J5}. ID_j is the group members' identity and $g^{r_i r_j}$ the secret key stored at both sides. HMAC(k,m) [22] is a pseudorandom function which hashes a message m using key k. The symbol "|" means concatenation. $K_{ij\text{-}e}$ is the encryption key, while $K_{ij\text{-}a}$ is used for message authentication. The third part of message M_{J5} for the new member contains the new group key GK_{new}, its version, the group key association GSA, which specifies the cryptographic algorithms and the security policies currently deployed for group communication, and the token holders' identity. These data are encrypted with the shared key SK determined during the authentication phase (see above). The fourth part contains the token, the token version VT, the neighbor's identity ID_{i+1}, the new members' identity ID_{n+1}, and its public DH value $g^{r_{n+1}}$. This part is protected under the new group key GK_{new}.

After having received M_{J5} the old group members can decrypt ID_i and N_i with the old group key GK_{old}. They can determine the channel key according to formula (1) and (2) and decrypt the new group key GK_{new}. The new group member decrypts the new group key using the shared key SK. Now all group members including the new participant P_{n+1} possess the new group key, the public DH values of the other members as well as their shared secrets. They are all in the same state so that the new token holder can refresh the group key when required.

Leave procedure. When a participant leaves the group the underlying group communication protocol informs the remaining members about the leaving. After that the token holder starts the key refreshment procedure. Figure 3 shows an example.

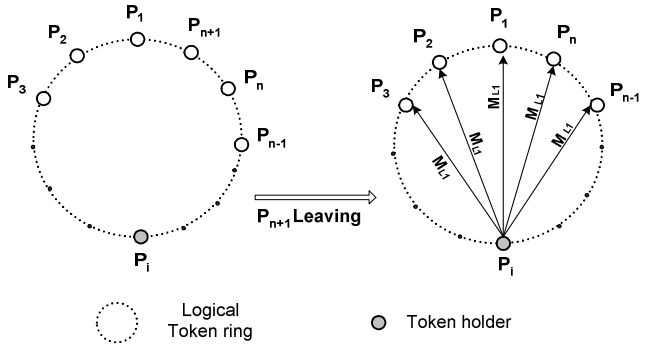

Fig. 3. Leave Procedure

We assume that participant P_{n+1} is leaving a group of n+1 members $P_1 \ldots P_{n+1}$. The token holder P_i generates a new group key GK_{new} and multicasts it in the leaving message M_{L1} to the remaining group members. M_{L1} has a similar structure like the join message M_{J5}:

$$M_{L1}(P_i \rightarrow P_1, P_2 \ldots P_n) : HDR, GK_{old}\{ID_i, N_i\}, K_{i1}\{VK, GK_{new}\}, \ldots, K_{in}\{VK, GK_{new}\}, \mathrm{GK}_{new}\{Token, VT, ID_{i+1}\}$$

It first encrypts the identity of the token holder ID_i together with a newly generated nonce N_i with the old group key. The new key GK_{new} and the actual key version *VK* are encrypted with the temporary channel keys for each remaining group member. These keys are derived according to formula (1) and (2). The token, the token version, and neighbor's identity ID_{i+1} are protected by the new group key. The leaving participant cannot take possession of GK_{new}, because it cannot reconstruct any of the temporary secure channels K_{i1}, $K_{i2}…K_{in}$ without knowing the shared secrets $g^{r_i r_1}, g^{r_i r_2}, …g^{r_i r_n}$ between the token holder P_i and the remaining members P_1, P_2,… P_j,…P_n ($j \neq i$). When receiving message M_{L1} the remaining group members can decrypt the new group key in the same way as described above for message M_{J5}. The group key refreshment is completed.

When the token holder leaves the group it first forwards the token to its neighbor and then starts the leave procedure. A host failure including that of the token holder is indicated by the underlying group communication layer. The group members equally update the group composition. In case of a member crash the token holder simply refreshes the group key. If the token holder crashes the neighbor in order is the next token holder by rule. It starts the leave procedure.

4 Performance analysis

This section evaluates the performance of TKD in comparison with other decentralized group key exchange protocols. We consider the key distribution protocol of Rodeh et al. and TGDH which are considered the most efficient group key distribution and agreement protocol, respectively. For the comparison, we apply the benchmarks of cryptographic algorithms from [23] to calculate the performance. The benchmarks are summarized in *Appendix.*

TKD comprises two different procedures: member authentication (message M_{J1}~ M_{J4}) and group key renewal (message M_{J5} or M_{L1}). Since the other two protocols do not possess a member authentication we only compare the cost for the key renewal.

4.1 Group key renewal

Group key renewal delay. A widely accepted criterion to evaluate the efficiency of group key management protocols is the group renewal delay. It refers to the duration of the key renewal, i.e. the time between the triggering of the procedure and the successful delivery of the key to each group member.

The group key renewal delay comprises the communication delay and the cryptographic computation delay. It is determined by the following formula:

$$D_{gkr} = \max(D_{cs} + D_{com} + D_{cr}) \quad (3)$$

where D_{gkr} is the group key renewal delay, D_{cs} the cryptographic computation delay of the sender, D_{com} the communication delay, and D_{cr} the cryptographic computation delay of the receiver. Here the sender stands for the *tree leader* in Rodeh's protocol, the *sponsor* in TGDH, and the *token holder* in TKD, respectively. The receiver corresponds to the participants including the new member and the subtree leader in the Rodeh protocol, and the participants including the new member in TGDH and TKD.

The **cryptographic computation delay** directly depends on the computation cost of the protocol. Table 1 summarizes the computation cost of the considered protocols by indicating the number of cryptographic operations they carry out. The comparison shows that the protocols use asymmetric and symmetric cryptographic operations differently. TGDH at the one edge applies more intensively asymmetric cryptographic computations while TKD at the other edge uses mainly symmetric operations. Since asymmetric cryptographic computations, as known, are much slower than the symmetric operations, the resulting total computation cost of TKD is lower than that of the other two protocols. For example, the computation cost of one DH agreement corresponds to approximately 5500 hash and symmetric encryptions of the group key (16 byte size) based on the benchmarks of cryptographic algorithms of [23]. Assuming a group size of 100 members, only about 400 hash and symmetric encryptions are required to renew the group key in TKD.

Table 1. Computation cost for the group key renewal

Protocols	Operation	Members	Computation cost				
			Asymmetric operations			Symmetric operations	
			DH agreement	RSA signature[2)]	RSA verification[2)]	Hash and encryption (16 Byte)	Hash and decryption (16 Byte)
Rodeh	Join	Tree leader	1	-	-	2	-
		New member	1	-	-	-	1
		Participants	-	-	-	-	1
	Leave	Tree leader	$\log_2 n$[1)]	-	-	$\log_2 n$	-
		Subtree leader	1	-	-	-	1
		Participants	-	-	-	-	1
TGDH	Join	Sponsor	2 $\log_2 n$	1	-	-	-
		New member	2 $\log_2 n$	-	1	-	-
		Participants	1…2$\log_2 n$	-	1	-	-
	Leave	Sponsor	2 $\log_2 n$	1	-	-	-
		Participants	1…2$\log_2 n$		1	-	-
TKD	Join	Token holder	-	-	-	2n+3	-
		New member	-	-	-	-	2
		Participants	1	-	-	-	3
	Leave	Token holder	-	-	-	2n+2	-
		Participants	-	-	-	-	2

Note: 1) n is the number of group members.
2) RSA signature in TGDH is used to support message authentication rather than member authentication [5].
3) The computation costs of Rodeh and TGDH listed in the table are their best case when the key tree is balanced. For an imbalanced key tree Rodeh and TGDH need more computation for a rekeying event.

Applying the benchmarks of the cryptographic algorithms from [23] to Table 1 we can now compute the cryptographic computation delay for the three protocols, i.e. $max(D_{cs}+D_{cr})$. The results are listed in Table 2. It shows that TKD causes less computation delay than the other two protocols for the join and leave procedure.

Table 2 Computation delay for group key renewal (ms)

	Rodeh	TGDH	TKD
Join	7.72	$4.93+15.44*\log_2 n$ [1)]	$3.86+(2n+6)*7*10^{-4}$
Leave	$3.86*(\log_2 n+1)+(\log_2 n+1)*7*10^{-4}$	$4.93+15.44*\log_2 n$	$(2n+4)*7*10^{-4}$

Note: 1) n is the number of group members.

The **communication delay** for a group key renewal depends on the number of communication rounds needed to complete the renewal procedure and on the duration of the communication rounds. For a fair comparison, we assume here that the three protocols run on top of the group communication protocol GCP [19]. According to [24] the delay D_{c1} for one communication round in a LAN setting can be estimated as follows:

$$D_{c1} = 8.71n-8.63+0.0015b \quad (4)$$

where n is the number of group members and b the size of the rekeying message in bytes. TKD and TGDH require only one communication round to accomplish the group key renewal for joining and leaving, whereas the Rodeh protocol needs two communication rounds each.

Group key renewal delay. Based on Table 2 and formula (4), we can now determine the group renewal delay using formula (3). The resulting delays for the join and the leave procedures are depicted in Figure 4 and 5.

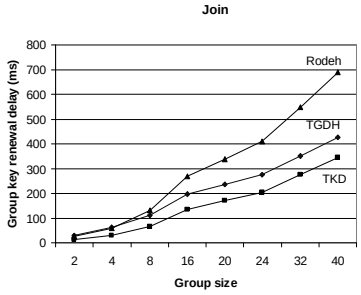

Fig. 4. Group key refreshment delay comparison for joining

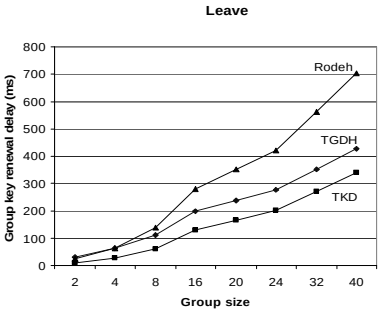

Fig. 5. Group key refreshment delay comparison for leaving

Inferred from formula (4) and the cited benchmarks of asymmetric algorithms in the appendix, the communication delay plays a larger role than the cryptographic computation delay in determining the group key refreshment delay. It is worth to mention that a reverse conclusion may be drawn, if the comparisons are based on lower computation power devices (e.g. PDAs, mobile phones) rather than on a modern computer platform (such as Pentium 4 platform used in our comparison). The lower the power of a device the larger the computation delays required by the asymmetric algorithms.

The comparison shows that TKD has a lower group key refreshment delay for both procedures than the Rodeh key distribution protocol and the most efficient key agreement protocol TGDH. The reason is that TKD needs less computation rounds and mostly uses symmetric cryptographic computations. The Rodeh protocol is the most expensive one, because it demands two communication rounds to accomplish the group key refreshment. Although also requiring only one communication round like TKD the TGDH protocol is the second expensive one, because it requires a lot of asymmetric cryptographic operations to generate the new group key for each member.

4.2 Communication overhead

The communication overhead depends on the message size of the protocol, i.e. how many bandwidth a protocol consumes. Many centralized group key distribution protocols such as LKH [5] apply a key tree structure, so that the group key server can update the group key using a message size of $O(log_2 n)$ symmetric keys. This is of particular significance for very large groups (e.g. one million members). Thus the rekeying message can be delivered in one packet. Some decentralized group key distribution protocols like Rodeh and TGDH follow the same principle to achieve a small rekeying message size, $O(log_2 n)$ symmetric keys for Rodeh and $O(log_2 n)$ asymmetric keys for TGDH. This is, however, achieved at the cost of two communication rounds in the Rodeh protocol and of $O(log_2 n)$ asymmetric cryptographic computations in TGDH. This is reason why they are slower than our scheme. In TKD the rekeying message size is $O(n)$ symmetric keys. For a group of 100 member peers, these are about 4 Kbyte. This can be transmitted without any problem in one UDP packet. Therefore bandwidth consumption for key renewal is not an issue for small group settings at all. In contrast, the group key renewal delay is the critical point for real-time applications.

4.3 Theoretical upper bounds of the group size

Finally we estimate the theoretical upper bounds of the group size of TKD. This estimation is made on two conditions: (1) the key renewal delay of TKD should fall below that of the compared protocols and (2) TKD accomplishes the group key renewal in one communication round.

The group key renewal delay of TKD, TGDH, and the Rodeh protocol can be determined based on Table 2 and formula (4) as follows:

$$D_{TKD} = 3.86 + (2n + 6) * 7 * 10^{-4} + 8.71n - 6.83 + 0.0015 * 36 * n \quad (5)$$

$$D_{TGDH} = 4.93 + 15.44 * \log_2 n + 8.71n - 6.83 + 0.0015 * 128 * \log_2 n \quad (6)$$

$$D_{Rodeh} = 7.72 + 2 * (8.71n - 6.83 + 0.0015 * 36 * \log_2 n) \quad (7)$$

Condition (1) can be expressed through the following two formulas:

$$D_{TKD} \leq D_{TGDH} \quad (8)$$

$$D_{TKD} \leq D_{Rodeh} \quad (9)$$

These formulas can be now used to determine the upper limit of the TKD group size. The solution for formula (8) is n=3420, whereas formula (9) remains true for any group size, i.e. TKD is always more efficient than the Rodeh protocol if TKD completes the key renewal in one communication round. Condition (2) means that the size of the rekeying messages is always less than the maximum size of an UDP packet (65536 bytes). Thus the upper bounds of group size can be determined by the following formula:

$$(n + 1) * (sk + h) + ak \leq 65536 \quad (10)$$

where n is the number of group members, while *sk*, *h*, and *ak* correspond to the size of the symmetric key, the hash value and the asymmetric key, respectively. Their corresponding typical values are 16 bytes, 20 bytes and 128 bytes. Formula (10) holds as long as n is smaller than 1815.

To sum up, TKD is more efficient related to the key renewal delay than other key exchange protocols as long as the group size does not exceed 1815 members.

5 Security demands

TKD fulfills important security demands. Due to space limitation we cannot give a detailed analysis of the security properties of TKD here. We sketch the most important aspects.

The protocol has to assure that nobody outside the group acquires the group key (*key authentication*). TKD assures this, because each invitee must be authenticated by a group member. Only if this authentication is successful, the invitee can join the group and obtain the group key. On the other hand, the invitee can convince itself by receiving their identities and public DH values with message M_{J3} that the received group key is of the expected group.

It has to be assured that earlier leaving members cannot access to any key generated later to further decrypt data exchanged (*forward confidentiality*). According to the leaving procedure described above the leaving participant can never access to the temporary secret channels, because they rely on the shared secrets between the token holder and the remaining members which are not accessible for him/her. Vice versa, a later joining member should never have access to any older key to decrypt previously exchanged data (*backward confidentiality*). This is achieved by never deliver-

ing the old group key to the joining member. The parts of message M_{J5} that can be decrypted by the new member do not contain the old key.

The protocol has further to assure that members after leaving a session are not being able to deduce the current key using their former keys (*collusion freedom*). TKD assures this by randomly generating new group keys each time the group composition changes, which do not depend of each other. A further desirable feature is to avoid that a compromised group key can lead to the disclosure of past keys (*perfect forward secrecy*). TKD achieves this by never using long-term credentials for encrypting the keys and avoiding any access to keys and key material of former sessions. Finally the protocol should not allow that a disclosure of past session keys could be used to compromise the current group key (*resistance to known key attacks*). TKD prevents such attack by never transmitting the shared DH secrets via the network so that an attacker is unable to access the temporary secrete channels used for the group key delivery.

Compared with the other mentioned key management protocols TKD is the only one which fulfills all these security requirements.

6 Final remarks

In this paper we have presented the group key distribution protocol TKD to support confidential meetings of small dynamic peer groups in the Internet, e.g. for business talks. The protocol approach is simple and straightforward using a token mechanism and mainly symmetric cryptographic operations for the group key renewal. This leads to a significantly lower key renewal delay compared to existing key distribution and key agreement protocols. It is especially appropriate for applications which except key management and encryption/decryption simultaneously run other time and resource consuming procedures such as media data decompression like in a peer-to-peer multiparty video conference. In addition, TKD introduces in contrast to others key management protocol an authentication based on IKEv2. TKD requires an underlying group communication protocol that supports virtual synchrony for group data consistency as well as dynamic group composition. There are several protocols proposed in literature which possess these properties.

We showed in the paper that a simple straightforward distribution approach is more efficient for small groups than a key tree based one. It provides a stable performance and needs less effort for its maintenance. Key tree based schemes like TGDH and the Rodeh protocol possess a fluctuating performance after many rekeying operations due to the unbalance of the key tree. To maintain a balanced key tree rebalance algorithms have to be applied which makes the protocols more complex and less practical.

TKD further fulfills important security demands like key authentication, forward and backward confidentiality, collusion freedom, and others. It is based on the well-studied protocol IKEv2 for member authentication and the well-known Diffie-Hellman problem (i.e. discrete logarithm problem) for the construction of the temporary secure channels. The security of TKD is achieved by carefully paying attention that group key material can be only accessed by current authenticated group members and by generating keys which do not depend of each other.

We are currently introducing TKD into the security architecture of the peer-to-peer multiparty video conference system BRAVIS [25] to allow confidential talks of closed group meetings in the Internet.

References

1. P. McDaniel, A. Prakash, and P. Honeyman: Antigone: A Flexible Framework for Secure Group Communication, CITI Technical Report 99-2, University of Michigan.
2. P. S. Kruus: A survey of multicast security issues and architectures. In: Proc. of the 21st National Information Systems Security Conference (NISSC), Oct. 1998.
3. S. Rafaeli and D. Hutchison: A Survey of Key Management for Secure Group Communication. ACM Computing Surveys 35 (2003) 3: 309-329.
4. H. Harney and C. Muckenhirn: Group Key Management Protocol (GKMP) Specification, July 1997, RFC 2094.
5. C. Wong, M. Gouda, and S. Lam: Secure group communication using key graphs. IEEE/ACM Transactions on Networking 8 (2000)1: 16-30.
6. M. J. Moyer, J. R. Rao and P. Rohatgi: Maintaining balanced key trees for secure multicast. Technical report, IETF, June 1999. draft-irtf-smug-key-tree-balance-00.txt.
7. Y. Kim, A. Perrig, and G. Tsudik: Tree-based Group Key Agreement. ACM Transactions on Information Systems Security (TISSEC) 7(2004)1: 60-96.
8. E. Rescorla: Diffie-Hellman Key Agreement Method. RFC 2631, June 1999.
9. M. Burmester and Y. Desmedt: A secure and efficient conference key distribution system. In Advances in Cryptology (EUROCRYPT'94), Springer LNCS 950, 1995, pp. 275-286.
10. M. Steiner, G. Tsudik, and M. Waidner: CLIQUES: A new approach to group key agreement. IEEE ICDCS, 1998, pp. 380-397.
11. Y. Kim, A. Perrig, and G. Tsudik: Simple and fault-tolerant key agreement for dynamic collaborative groups. In: S. Jajodia (ed.): 7th ACM Conference on Computer and Communications Security, Athens, Greece, Nov. 2000, ACM Press, pp. 235–244.
12 L. Dondeti, S. Mukherjee, and A. Samal: Disec: A distributed framework for scalable secure many-to-many communication. In: Proc. of the 5th IEEE Symposium on Computers and Communications (ISCC), July 2000.
13. O. Rodeh, K. P. Birman, D. Dolev: Optimized Group Rekey for Group Communication Systems. In Symposium Network and Distributed System Security (NDSS), San Diego, California, Feb. 2000, pp. 39-48.
14. G. V. Chockler, I. Keidar, and R. Vitenberg: Group communication specifications: A comprehensive study. ACM Computing Surveys 4 (2001): 427-469.
15. B. Whetten, T. Montgomery, and S. Kaplan: A High Performance Totally Ordered Multicast Protocol. International Workshop on Theory and Practice in Distributed Systems, Springer LNCS 938, pp. 33-57, 1994.
16. D. A. Agarwal: Totem: A Reliable Ordered Delivery Protocol for Interconnected Local Area Networks, Ph.D Thesis, University of Santa Barbara, Dec 1994.
17. K. Birman, R. Constable, M. Hayden, C. Kreitz, O. Rodeh, R. Van Renesse, W. Vogels: The Horus and Ensemble Projects: Accomplishments and Limitations. In: Proc. of the DARPA Information Survivability Conference & Exposition (DISCEX '00), Hilton Head, South Carolina, 2000.
18. Y. Amir, C. Danilov, and J. Stanton: A low latency, loss tolerant architecture and protocol for wide area group communication. In: Proc. 30th IEEE FTCS, June 2000.

19. M. Zuehlke, and H. Koenig: A Signaling Protocol for Small Closed Dynamic Multi-peer Groups. In Z. Mammeri and P. Lorenz (eds.): High Speed Networks and Multimedia Communications (HSNMC 2004). Springer-Verlag, Berlin, Heidelberg 2004, pp. 973 – 984
20. D. Harkins and D. Carrel: The Internet Key Exchange (IKE), RFC 2409, Nov. 1998.
21. C. Kaufman, Internet Key Exchange (IKEv2) Protocol, draft-ietf-ipsec-ikev2-17.txt, September 2004.
22. H. Krawczyk, M. Bellare, and R. Canetti: HMAC: Keyed-Hashing for Message Authentication, RFC 2104, February 1997.
23. Crypto++ 5.2.1 Benchmarks. http://www.eskimo.com/~weidai/benchmarks.html
24. M. Zuehlke: Distributed organized multiparty video conference for closed group in the Internet. Ph.D thesis, Brandenburg University of Technology Cottbus, Department of Computer Science, May 2004.
25. The BRAVIS video conference system. http://www.bravis.tu-cottbus.de.
26. S. Chanson, A. Hui, E. Siu, I.Beier, H. Koenig, and M. Zuehlke: OCTOPUS-A Scalable Global Multiparty Video Conferencing System. In: Proc. of the 8th International IEEE Conference on Computer Communications and Networks (IC3N'99), Boston, 1999, pp.97-102

Appendix: Benchmarks of crypto operations

The speed benchmarks for the cryptographic algorithms used in this paper are listed in the following table which is adopted from [23]. These results are achieved on a Pentium 4 2.1 GHz processor under Windows XP.

	Symmetric crypto operations		Asymmetric crypto operations		
Algo-rithms	SHA-1	AES (128-bit key)	RSA 1024 Signature	RSA 1024 Verification	DH 1024 key Agreement
Speed	68 Mbyte/s	61 Mbyte/s	4.75 ms/operation	0.18 ms/operation	3.86 ms/operation