Contributory Group Key Agreement Protocols, Revisited for Mobile Ad-Hoc Groups[*]

Mark Manulis[†]
Horst-Görtz Institute, Ruhr-University of Bochum
IC4/157, D-44801 Bochum, Germany
mark.manulis@rub.de

Abstract

Security of various group-oriented applications for mobile ad-hoc groups requires a group secret shared between all participants. Contributory group key agreement (CGKA) protocols, originally designed for peer groups in local- and wide-area wired networks, can also be used in ad-hoc scenarios because of the similar security requirements and trust relationship between participants that excludes any trusted central authority (e.g., a group manager) from the computation of the group key. We revise original protocols from the perspective of the mobile ad-hoc communication, classify mobile ad-hoc groups based on the performance of involved mobile devices, specify trust relationship between participants, propose further optimizations to original protocols to achieve better communication, computation and memory complexities.

1. Introduction

Consider a group of people who wish to establish secure ad-hoc communication using their mobile devices. The group is called *dynamic* if it allows to add and delete participants during the communication session; otherwise the group is *static*. Dynamic events should be handled without any risks to the communication security. Participants may be equipped with mobile devices of different types, e.g., laptops, PDAs or phones. Thus, performance constraints may differ from device to device. We call an ad-hoc group *heterogeneous* if its members are equipped with different kinds of devices, and *homogeneous* if devices have similar performance properties. Note that our notion of heterogeneity in ad-hoc groups refers to the performance of the devices, whereas heterogeneity of ad-hoc networks is frequently used in the literature with respect to the underlying infrastructure. Homogeneous ad-hoc groups may be, for example, meetings of employees of a company that are equipped with equal devices according to some organizational policy. In the contrary, heterogeneous groups are meetings, like conferences, auctions, elections, where participants and involved devices are unpredictable. The task of securing the communication reduces to the problem of the shared secret key establishment among all participants, and its maintenance over the dynamic events. The group key management must consider different ad-hoc communication requirements.

In this paper we consider *contributory group key agreement* (CGKA) protocols: BD [3], CLIQUES [13], STR [7] and TGDH [8], that were originally designed for local- and wide-area wired networks. These protocols allow participants to compute the group key as a function of their personal contributions, and can be applied in mobile ad-hoc scenarios because of the similar security requirements and trust relationship between participants. Essential is the absence of a trusted central authority (e.g., a group manager or a key server) that is actively involved in the computation of the group key. Since these CGKA protocols have been designed for local- and wide-area wired networks we optimize them with respect to the communication, computation and memory constraints in a mobile ad-hoc environment, and discuss their suitability for static and dynamic, homogeneous and heterogeneous ad-hoc groups. Some parts of Section 7 may be considered as a survey of existing CGKA protocols from the perspective of mobile ad-hoc networks based on the requirements that have not been taken previously into account.

2. Related Work

Amir *et. al.* ([1]) have compared performance of these protocols for local- and wide-area wired networks. Their analysis includes the total number of required rounds and messages, and serial (i.e., operations that can be computed by members in parallel are counted as a single operation) computation costs for heavy operations, like modular exponentiations. Their work does not take into account some special requirements that have to be considered in ad-hoc networks. Neither [1] nor any other work describes the

∗ This is a full version of the paper appeared in Proceedings of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2005), International Workshop on Wireless and Sensor Networks Security (WSNS 2005), November 07-10, Washington, USA. © IEEE Computer Society 2005

† This work is supported by the European Commission through IST-2002-507932 ECRYPT.

memory complexity of the protocols. We close this gap comparing the size of data that has to be stored per device. Another point of interest is the total size of sent messages, because of the limited bandwidth in mobile networks. This is not covered in their work either. Serial computation costs say nothing about the actual costs that a certain device has to bear. Especially for heterogeneous groups, the knowledge of the exact computation costs per device is essential. Bhaskar ([2]) provides exact computation costs for CLIQUES and STR, but only average costs for TGDH. He compares original protocols without considering any possible optimizations that may lead to a significant performance enhancement as we show throughout this paper.

3. Our Contribution

The main contributions of this paper are: classification of homogeneous and heterogeneous ad-hoc groups based on the performance quantification presented in [10], definition of additional performance requirements, elimination of redundancy and performance enhancement of the described CGKA protocols for mobile ad-hoc scenarios, exact analysis of the computation and memory costs per device, and of the total size of sent messages, and discussion of suitability of optimized protocols for static and dynamic homogeneous and heterogeneous groups.

4. Mobile Ad-Hoc Group Communication

4.1. Mobile Devices

In order to distinguish between homogeneous and heterogeneous groups it must be possible to distinguish between performances of the mobile devices (i.e., to say that one device is more or less powerful than another). Intuitively, laptops are more powerful than PDAs, and PDAs are more powerful than mobile phones. *Performance ratio* parameter $\mu_i \in \mathbb{R}$ introduced in [10] allows to quantify the performance of a mobile device M_i (for simplicity the same notation is used for members and their devices). It is defined as a value returned by a benchmarking function f which takes as input the hardware parameters of M_i, such as CPU clocks, memory capacity and battery power consumption, and performs some network and cryptographic application specific operations. Let $\mathcal{M}$ be a set of mobile devices. For $M_i, M_j \in \mathcal{M}$, M_i is more powerful than M_j if $\mu_i > \mu_j$. The list $P = (M_1, \dots, M_{|\mathcal{M}|})$ is called a *performance ratio order* if $\mu_i \geq \mu_{i+1}$ holds for any $M_i, M_{i+1} \in \mathcal{M}$. Note that given a position i it is possible to reveal the corresponding device M_i and its performance ratio μ_i.

4.2. Mobile Ad-Hoc Groups

In this section we distinguish between various kinds of mobile ad-hoc groups.

Definition 1 *Let $\mathcal{G}$ be a mobile ad-hoc group, P a performance ratio order of n involved mobile devices, and $\epsilon \in \mathbb{R}$. $\mathcal{G}$ is called homogeneous if $\forall \mu_i, \mu_j \in P : |\mu_i - \mu_j| \leq \epsilon$, and heterogeneous if $\exists \mu_i, \mu_j \in P : |\mu_i - \mu_j| > \epsilon$. The value ϵ is called a limit of homogeneity.*

Note that ϵ can be specified in advance with respect to the performed application measurements on different kinds of mobile devices.

Mobile ad-hoc groups can be either static or dynamic. In *static* groups the initial number of participants remains unchanged during the whole communication period. In *dynamic* groups we distinguish between additive and subtractive events. *Additive* events are *join* (new participant has to be added to the group) and *merge* (merging of multiple groups to a single group). *Subtractive* events are *leave* (a current participant has to be excluded from the group) and *partition* (splitting of the group into multiple subgroups). A dynamic event can be either *explicit* if it is triggered by the application or *implicit* if it occurs unexpectedly (e.g., network failure).

5. Model

In this section we describe communication and security models for CGKA protocols in mobile ad-hoc networks.

5.1. Communication

The CGKA protocols require from the underlying group communication platform to be *public* (note that messages that are broadcasted over this channel can be intercepted by a passive adversary), and *reliable*, i.e., all messages reach their destination after being sent, and the order of sent messages is preserved. Reliability in ad-hoc networks can be achieved using reliable multicast protocols like RDG [9].

5.2. Security

Trust Relationship In a mobile ad-hoc group there is no trusted central authority that is actively involved in the computation of the group key, i.e., all participants have equal rights during the computation process. We emphasize this by definition of the *verifiable trust relationship* which should be provided by a CGKA protocol.

Definition 2 *A verifiable trust relationship consists of the following two requirements:*

1. *Group members are trusted not to reveal the group key or secret values that may lead to its computation to any other party, and*
2. *Group members must be able to verify the computation steps of the CGKA protocol.*

Authentication All CGKA protocols require authentic communication channels in order to prevent impersonation attacks. The authentication of messages can be achieved, for example, with digital signatures and certified public keys. Every participant M_i should have a certificate for its public key *pkey*$_i$ and use its secret key *skey*$_i$ to sign own protocol messages. Note that conventional PKI techniques with a trusted certification authority may not be available in a mobile ad-hoc network because of dynamics and missing infrastructure. Management of public key certificates in ad-hoc networks is currently a hot research topic, e.g., [4]. We assume that an appropriate management mechanism for public key certificates is available, so that participants can authenticate their messages. In the description of protocols we omit indication of the authentication since it is common for all protocols.

Requirements All CGKA protocols analysed in this paper fulfill the following security requirements from [8]: *computational group key secrecy* (it must be computationally infeasible for a passive adversary to discover any secret group key), *decisional group key secrecy* (it must be computationally infeasible for a passive adversary to distinguish any bits of the secret group key from random bits), *forward secrecy* (any passive adversary being in possession of a subset of old group keys must not be able to discover any subsequent group key), *backward secrecy* (any passive adversary being in possession of a subset of subsequent group keys must not be able to discover any preceding group key), and *key independence* (any passive adversary being in possession of any subset of group keys must not be able to discover any other group key). Additionally, [10] introduces a performance requirement that is specific to homogeneous and heterogeneous mobile ad-hoc groups: *cost fairness* (computation, communication and memory costs of the CGKA protocol must be distributed between mobile devices considering their performance ratios). Intuitively, it means that protocol costs are distributed uniformly in homogeneous, and non-uniformly in heterogeneous groups. Cost fairness immediately implies a security requirement, called *performance honesty*, that is no participant should be able to cheat on the performance ratio of its device, e.g., to pretend that it has a smaller performance ratio than it really does in order to save own protocol costs at the expense of other participants. Note that in case of performance honesty the adversary is active (i.e., a participant of the protocol), whereas in security requirements described above it is passive. According to [10] one possibility to achieve performance honesty is to use a tamper-resistant (trusted) hardware component[1] that stores an authentic performance ratio of the device, such that the user gains access to the computed group key only if the trusted component authenticates the claimed performance ratio (*property-based sealing*).

1 for example, the trusted platform module (TPM) proposed by the TCG

6. Efficiency with ECC

All CGKA protocols apply public key cryptography. Although it is costly there are no other techniques to agree on a key over a public channel. Therefore, in order to reduce computation costs we switch to *elliptic curve cryptography* (ECC) whose computation and communication costs are much smaller due to the smaller key sizes (≈ 160 bits). We can switch to ECC because all operations in the analysed protocols can be also performed in groups of points of elliptic curves defined over finite fields. Note that original protocols are described in the group $\mathbb{Z}_p^*$. It may seem that the mapping to ECC is mostly mechanical, however, several strong requirements as shown below have to be considered; otherwise the risk is large that protocols become insecure or that the computation of the group key fails because of the mathematical inconsistency. Mapping to ECC is a significant part towards a better efficiency of CGKA protocols when used in mobile ad-hoc groups. Let E be an elliptic curve over a finite field $\mathbb{F}_q$, such that $\mathbb{F}_q$ is either prime (q is a prime) or binary ($q = 2^m$, $m \in \mathbb{N}$) field. $E(\mathbb{F}_q)$ denotes a commutative group of points in E. Let $G \in E(\mathbb{F}_q)$ be a point with high prime order t that devides $q-1$. G generates a subgroup of $E(\mathbb{F}_q)$ denoted $<G>= \{O, G, 2G, \ldots, (t-1)G\}$, where O is the point of infinity. We remark that all computations in the optimized protocols are done in $<G>$. Some protocols require to map a point $Q \in E$ to an integer in the range $[1, \ldots, q-1]$. The most natural way is to map Q to its x-coordinate. We suggest to use the following function $map : E(\mathbb{F}_q) \to \mathbb{N}$ defined in [12]: if $q = p$ and p is prime then $map(Q) = (Q)_x$, else if $q = 2^m$, $m \in \mathbb{N}$, and $(Q)_x = (a_{m-1} \ldots a_1 a_0)$ with $a_i \in \{0, 1\}$ then $map(Q) = \sum_{i=0}^{m-1} 2^i a_i$.

7. Optimized CGKA Protocols

In this section we describe and optimize Burmester-Desmedt (BD) [3], CLIQUES [13], STR [7] and TGDH [8] protocols with respect to the requirements of mobile ad-hoc communication and performance limitations of devices. Due to space limits we describe only the setup protocols, and mention general ideas for the handling of dynamic events.

7.1. μBD

Original Burmester-Desmedt (BD) protocol ([3], [6]) arranges members in a ring structure, such that any member M_i, $i \in \{1, \ldots, n\}$ knows its neighbours: M_{i-1} and M_{i+1} (if $i = 1$ then M_{i-1}=M_n). Our elliptic curve equivalent μBD is given in Figure 1. All members compute the same group key K. BD is stateless, thus current group members have to restart the protocol after any new dynamic event. (μ)BD do not provide verifiable trust relationship, since no other group member can verify the correctness of the broadcasted X_i (note, at least two members have to cooperate).

- M_i selects random $r_i \in_{\mathcal{R}} \{1, \ldots, t-1\}$, computes and broadcasts $Z_i = r_i G$.
- M_i computes and broadcasts $X_i = r_i(Z_{i+1} - Z_{i-1}) = (r_i r_{i+1} - r_i r_{i-1})G$.
- M_i computes $K = n r_i Z_{i-1} + (n-1)X_i + \ldots + X_{i+n-2} = (r_1 r_2 + r_2 r_3 + \ldots + r_{n-1} r_n)G$.

Figure 1. μBD Setup

7.2. μCLIQUES

Original CLIQUES ([13]) is a CGKA protocol suite that arranges group members in a list structure $(M_1, \ldots, M_n)$. The protocol specifies a role of the controller that collects contributions of other group members, adds own contribution, and broadcasts information that allows all members to compute the group key. We stress that this role is temporary and does not mean a trusted central authority whose existence in the group was excluded. The choice of the controller depends on the dynamic event and the current list structure. CLIQUES mixes unicast and broadcast communication to achieve a better communication performance, since unicast communication requires less costs. Our elliptic curve equivalent μCLIQUES is given in Figure 2 with M_n as controller. In additive events new members are appended to the end of the list. To achieve key independence the controller changes its random value r'. Last appended member becomes a controller for the next additive event. The new group key is computed in the same manner as in the setup protocol, except for the difference that the computation process starts from the controller's position in the list. In subtractive events the set of leaving members $\mathcal{L}$ is

- $1 \le i \le n-2$: M_i selects random $r_i \in_{\mathcal{R}} \{1, \ldots, t-1\}$, and unicasts $Z_i = r_i Z_{i-1}$ to M_{i+1}. (note, $Z_1 = r_1 G$)
- M_{n-1} selects random $r_{n-1} \in_{\mathcal{R}} \{1, \ldots, t-1\}$, and broadcasts $Z_{n-1} = r_{n-1} Z_{n-2}$.
- M_i sends $X_i = Z_{n-1}/r_i$ to M_n.
- M_n broadcasts $\mathcal{S} = \{S_i = r_n X_i | 1 \le i \le n\}$.
  M_i computes $K = r_i S_i = r_1 r_2 \ldots r_n G$ with $S_i \in \mathcal{S}$.

Figure 2. μCLIQUES Setup

deleted from the list. The controller that is the most recent remaining member chooses new random value r' and computes $\mathcal{S}' = \{S'_i = r' S_i | 1 \le i \le n \wedge i \notin \mathcal{L}\}$. Upon receiving $\mathcal{S}'$ other members compute the new group key as in the last step of the setup protocol. If the controller is a leaving member then any other member can take over its role, assuming it has saved the previous set $\mathcal{S}$. (μ)CLIQUES do not provide verifiable trust relationship, because no other member can check whether values Z_i or X_i forwarded by M_i, or the set $\mathcal{S}$ broadcasted by the controller are correctly built.

7.3. μSTR

Original STR ([7]) is a CGKA protocol suite that arranges members in a binary tree structure from Figure 3. The tree has two kinds of nodes: leaf and internal nodes. An internal node IN$_i$ has two children: a lower internal node IN$_{i-1}$ and a leaf node LN$_i$. (IN$_1$ = LN$_1$ is the only exception). In the following we describe the structure of the group key in ECC. Each LN$_i$ is associated with member (device) M_i and contains its secretly chosen *session random* r_i. Its public version is $R_i = r_i G$. Each IN$_i$ is associated with a secret key k_i and its public counterpart $K_i = k_i G$. Every

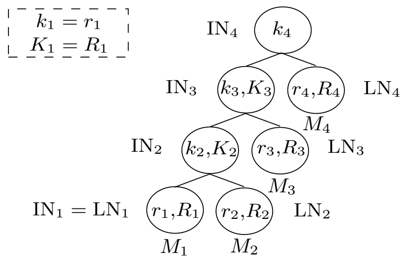

Figure 3. STR binary tree ($n = 4$)

$k_i = r_i k_{i-1} G$, $i > 1$ (note that $k_1 = r_1$) is computed using tree-based Diffie-Hellman key exchange method [8] in two different ways: $k_i = map(r_i K_{i-1})$ or $k_i = map(k_{i-1} R_i)$. Since k_i has to be an integer in order to compute k_{i+1}, but values $(r_i K_{i-1})$ and $(k_{i-1} R_i)$ are points in E, the point-to-integer mapping function map is used. The secret group key $K = k_n$ can be computed by any member M_i that knows K_{i-1} and all R_j for all $1 < i < j \le n$. We present μSTR in Figure 4. As in original STR it defines the role of the sponsor (similar to the controller in CLIQUES) that is

- M_i selects random $r_i \in_{\mathcal{R}} \{1, \ldots, t-1\}$, computes and broadcasts R_i.
- M_1 and M_2 compute $(k_2, \ldots, k_n)$. M_1 computes and broadcasts $(K_1, \ldots, K_{n-1})$.
- $M_i, i \neq \{1, 2\}$ computes $(k_i, \ldots, K = k_n)$.

Figure 4. μSTR Setup

temporary and can be assigned to different members on occured dynamic events depending on the current tree structure. The sponsor reduces the communication overhead as it performes some operations on behalf of the group. We stress that the sponsor is not a central authority. (μ)STR provide verifiable trust relationship because every broadcasted public key can be verified by at least one other participant, e.g., K_i can be computed by members M_i and M_j for all $j < i$. In additive events new members are added on top of the tree and sponsor M_s is the highest-indexed member in the initial tree. It changes own session random r'_s, computes changed secret keys k_i and public keys K_i for all $i \ge s$, and broadcasts updated tree with all public keys and public session randoms.[2] In subtractive events leaving members are removed from the tree and sponsor M_s is the member associated in the initial tree with the leaf node located directly below the leaf node of the lowest-numbered leaving member. Its computations are similar to those in additive events.

Optimization Original protocol requires that the sponsor M_s broadcasts the updated tree with all public keys and public session randoms. A closer look shows that some broadcasted data is redundant. Every member M_i, $i > 1$ must save only public key K_{i-1}, public session randoms R_j and secret keys k_j for all $j \geq i$ in order to be able to update the group key and be prepared to take over the sponsor's role in any further dynamic event. Thus, M_s has to broadcast only changed values K_j and R_j, $j \geq s$, since unchanged values are already known. These modifications reduce the size of broadcasted messages and the size of stored data in μSTR.

Modification for Heterogeneous Groups The protocol μSTR-H presented in [10] is a modification of μSTR for heterogeneous ad-hoc groups. μSTR-H distributes costs non-uniformly between all participants based on the performance ratio order P. The observation behind the μSTR-H is that members located deeper in the tree perform more computations and save more data than higher-located members. All members compute P and maintain it upon dynamic changes. The management policy of P ensures that more powerful devices are inserted below less powerful devices, and have, therefore, to bear higher computation and memory costs as required by the cost fairness. New devices are added into the tree on positions according to P, and not simply on top of the tree as in μSTR.

7.4. μTGDH

Original TGDH ([8]) is a CGKA protocol suite that arranges members in a binary tree structure from Figure 5. The tree is kept balanced, i.e., paths from leaf nodes up to the root contain equal number of intermediate nodes.

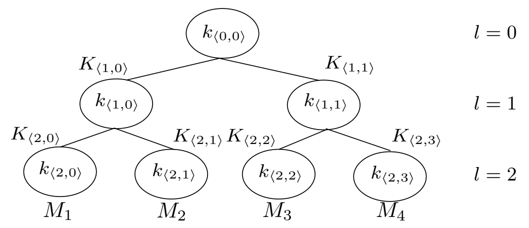

Figure 5. TGDH binary tree ($n = 4$)

Each node $\langle l, v\rangle$ is associated with a secret key $k_{\langle l,v\rangle}$ and a public key $K_{\langle l,v\rangle} = k_{\langle l,v\rangle}G$. Secret keys of leaf nodes are chosen and kept secret by associated members. Every $k_{\langle l,v\rangle}$, $0 \leq v \leq 2^l - 1$, $0 \leq l \leq h$ where h is the height of the tree, is computed using Diffie-Hellman key exchange between nodes $\langle l+1, 2v\rangle$ and $\langle l+1, 2v+1\rangle$ either as $k_{\langle l,v\rangle} = map(k_{\langle l+1,2v\rangle}K_{\langle l+1,2v+1\rangle})$ or $k_{\langle l,v\rangle} = map(k_{\langle l+1,2v+1\rangle}K_{\langle l+1,2v\rangle})$. The secret group key $K = k_{\langle 0,0\rangle}$ can be computed by any member M_i if it knows all public keys $K_{\langle l,v\rangle}$ in the tree. Although original description of TGDH does not specify the setup procedure, we derive the protocol in Figure 6 with respect to the group key structure and techniques of other protocols of the suite. In the setup protocol the sponsor of the (sub)tree is always the rightmost member. In additive events new member (or the sponsor of the merged group) sends own (group's) public key(s). Members determine the insertion node of the new member (or merged group's tree) that is the rightmost node that does not increase the height of the tree. Sponsor M_s of the event that is the rightmost member of the updated tree, changes own contribution $k_{\langle l_s,v_s\rangle}$, computes all new secret keys and public keys in its path up to the root, and broadcasts the updated tree with all public keys so that other members can update the tree and the group key upon the sponsor's message. In subtractive events leaf nodes of leaving members are deleted from the tree. In case of leave the sponsor M_s is the rightmost member of the subtree rooted at the leaving member's sibling node. Computations of M_s are equal to those in additive events. In case of partition the leave protocol is performed for every partitioned member in parallel. Thus, there may exist multiple sponsors that compute new secret and public keys as far up the tree as possible before they broadcast corresponding trees including the public keys. Partition protocol may take several rounds, since the computation of the group key may be blocked until the required public keys are broadcasted. (μ)TGDH provide verifiable trust relationship, because every broadcasted public key $K_{\langle l_s,v_s\rangle}$ can be verified by every group member in the subtree rooted at node $\langle l_s, v_s\rangle$.

1. M_i selects random $k_{\langle l_i,v_i\rangle} \in_{\mathcal{R}} \{1, \ldots, t-1\}$, computes and broadcasts $K_{\langle l_i,v_i\rangle}$, sets $l := l_i - 1$ and $v := \lfloor v_i/2 \rfloor$.
2. M_i updates the tree structure, computes secret key $k_{\langle l,v\rangle}$, and public key $K_{\langle l,v\rangle}$. The sponsor of the (sub)tree rooted at node $\langle l, v\rangle$ broadcasts $K_{\langle l,v\rangle}$.
3. Members repeat steps 2 and 3 with $l := l-1$ and $v := \lfloor v/2 \rfloor$ until every member can compute the group key $K = k_{\langle 0,0\rangle}$.

Figure 6. μTGDH Setup

Optimization Original TGDH can be optimized similarly to STR. Every member requires only public keys of all sibling nodes in its path up to the root to compute the group key. However, members still have to save the tree structure

2 In the original description of join it is not specified that a new member receives all public keys and public session randoms. However, in leave protocol the sponsor has to broadcast the whole tree with all these values (denoted there as BT_s). Thus, in case that a new member becomes the sponsor of the leave event it must also know all public keys and public session randoms of the current tree.

to be able to update it after dynamic events. Thus, members save the tree structure with required public keys. The size of the sponsor's message can be reduced, if it broadcasts updated tree including only changed public keys.

7.5. Security Analysis

Original security proofs of the described protocols are given for BD in [6], for CLIQUES in [13], for STR in [7], and for TGDH in [8]. All proofs build reductions from the attacks against security requirements of the protocol to attacks against well-known cryptographic assumptions. Our optimizations do not change the computation process of the group key in either of the described protocols. In all protocols we map mathematical operations from $\mathbb{Z}_p^*$ to $<G>$, i.e., switch to ECC according to Section 6. ECC does not bring any security risks since all cryptographic assumptions that hold in $\mathbb{Z}_p^*$ and are used in the original proofs, also hold in $<G>$ yielding that all optimized protocols remain secure. The following cryptographic assumptions are used in original proofs: Discrete Logarithm (DL) assumption (i.e., given a generator g of a multiplicative cyclic prime order group $\mathbb{G}$ and $g^a \in \mathbb{G}$, it is hard to compute a); Computational Diffie-Hellman (CDH) assumption (i.e., given g, g^a, $g^b \in \mathbb{G}$, it is hard to compute g^{ab}); Decisional Diffie-Hellman (DDH) assumption (i.e., given g, g^a, g^b, $g^c \in \mathbb{G}$, it is hard to decide whether $g^c = g^{ab}$). According to [11] the ECC counterparts of DL and CDH assumptions are hard in $<G>$ for all types of elliptic curves. However, the hardness of the DDH assumption could only be proven for special types of elliptic curves, i.e., non-supersingular and non-trace-2 elliptic curves as described in [5]. Therefore, only these special curves should be chosen for the implementation.

7.6. Complexity Analysis

Table 1 provides communication, computation and memory costs of the optimized protocols. We consider one protocol round as over if members have to wait for missing data to continue with the computation of the group key. Columns U and B represent the total number of unicast and broadcast messages, respectively. The message size column gives the total size of sent messages in $\log q$-bits where q is the parameter of the finite field F_q (in practice $q \approx 160$ bits). Computation costs specify the total number of scalar-point multiplications per member based on member's index (position) in the group. This creates a basis for the suitability analysis of the protocols for homogeneous and heterogeneous groups. The memory costs column specifies the size of data that a device has to store in order to handle dynamic events. The following notations are used: n - initial group size, i - updated index (position) of M_i, s - updated index (position) of the sponsor, m - size of the merging group, p - number of leaving (partitioned) members, h - height of the TGDH tree (note $h = \lceil \log n \rceil$), l_i (l_s) - updated level of member's M_i (sponsor's M_s) node in TGDH tree (note, $l_i, l_s \in \{0, \dots, h\}$), l_{s_j} (l'_{s_j}) - updated (initial) level of sponsor's M_{s_j} node in TGDH tree in merge and partition protocols, $T_{\langle l_{s_j}, v_{s_j} \rangle}$ - subtree that represents the initial merging group with sponsor M_{s_j}, M_{s_r} - the rightmost sponsor in μTGDH partition, s_j^* - index of sponsor M_{s_j} whose level l_{s_j} is maximal compared to other sponsors in μTGDH merge.

Communication Obviously, μSTR provides best communication efficiency concerning the total number of rounds and sent messages. The total messages size in case of join is constant, in case of merge depends on the number of merging members, and in other cases scales linearly with the sponsor's position, varying between 1 and n. Compared to μSTR the size of μTGDH messages scales linearly with the level of sponsor's node l_s, which varies between 0 and $h = \lceil \log n \rceil$. Thus, in some cases μTGDH may require less communication bandwidth than μSTR.

Computation μBD protocol requires only 3 scalar-point multiplications (we do not count additional $n - 1$ multiplications with a small integer whose costs may become non-negligible for large n). From all protocols that were designed to handle dynamic events we point out μCLIQUES and μTGDH. μCLIQUES requires a constant number of multiplications for all members except for the sponsor. Significant drawback is that the number of sponsor's multiplications scales linearly in the number of group members. In μTGDH the number of multiplications performed by M_i is given by the function f (note $f(i, s) \leq \mathit{min}(l_i, l_s)$), and can be approximated by $O(\log n)$. Notable is also that in μSTR and μSTR-H the number of multiplications per member is proportional to its node's position in the tree. This allows non-uniform distribution of costs as required in heterogeneous groups.

Memory μBD is stateless and requires, therefore, from group members to save only the group key. However, the protocol has to be restarted to update the group key after occuring dynamic events. The handling of dynamic events by other CGKA protocols requires from members to save some auxiliary information. In μCLIQUES all members have to save equal amount of information (i.e, $(n + 1) \log q$ bits), regardless of their position in the group. In μTGDH required memory space depends on the level of member's node l_i, which varies between 0 and $h = \lceil \log n \rceil$. Since the tree management policy of μTGDH tries to keep the tree balanced most members have to save $\lceil \log n \rceil$ keys (i.e, $\lceil \log n \rceil \log q$ bits), whereas in μSTR and μSTR-H the number of keys that a member has to save scales linearly with his position in the tree and may, therefore, vary between 4 and $2n$ keys (i.e, between $4 \log q$ and $2n \log q$ bits). This is essential for heterogeneous groups where less-powerful devices are assigned to the lower nodes and have to save, therefore, less data.

Table 1: Computation, Communication and Memory Costs of Optimized CGKA Protocols

CGKA Protocol		Communication				Computation	Memory
		Rounds	U	B	Message size	Scalar-Point Multiplications	Saved data
μBD	S	2	0	$2n$	$2n$	3	1
μCLIQUES	S	$n+1$	$2n-3$	2	$3n-2$	$i<n-1$: 3 $i=n-1$: 2, $i=n$: n	$n+1$
	J	2	1	1	$2n+2$	$i=n, i=n+1$: $n+1$ $i<n$: 1	
	L	1	0	1	$n-1$	$i=s$: $n-1$ $i\neq s$: 1	
	M	$m+1$	m	1	$\frac{m^2+5m+4n+2}{2}$	$i<n$: 1 $n\leq i\leq n+m$: $i+2$	
	P	1	0	1	$n-p$	$i=s$: $n-p$ $i\neq s$: 1	
μSTR	S	2	0	$n+1$	$2n-2$	$i=s$: $2n-1$ $i\neq s$: $n-i+2$	$i=1$: $2n$ $i>1$: $2(n-i+2)$
	J	1	0	2	3	$i=s$: 4 $i\neq s$: 2	
	L	1	0	1	$n-s$	$i<s$: $n-s$ $i=s$: $2(n-s)$ $i>s$: $n-i$	
	M	2	0	3	$2m$	$i<s$: $m+1$ $i=s$: $2(n+m-s+1)$ $i>s$: $n+m-i+1$	
	P	1	0	1	$n-p-s+1$	$i<s$: $n-p-s+1$ $i=s$: $2(n-p-s+1)$ $i>s$: $n-p-i+1$	
μSTR-H	S	2	0	$n+1$	$2n-2$	$i=s$: $2n-1$ $i\neq s$: $n-i+2$	$i=1$: $2n$ $i>1$: $2(n-i+2)$
	J	1	0	2	$2n-2s+3$	$i<s$: $n-s+2$ $i=s$: $2(n-s+2)$ $i>s$: $n-i+2$	
	L	1	0	1	$n-s$	$i<s$: $n-s$ $i=s$: $2(n-s)$ $i>s$: $n-i$	
	M	2	0	3	$2n+2m-s+1$	$i<s$: $n+m-s+1$ $i=s$: $2(n+m-s+1)$ $i>s$: $n+m-i+1$	
	P	1	0	1	$n-p-s+1$	$i<s$: $n-p-s+1$ $i=s$: $2(n-p-s+1)$ $i>s$: $n-p-i+1$	
μTGDH	S	h	0	$2n-2$	$2n-2$	v_i even: l_i+1 v_i odd: l_i+k	$2(l_i+1)$
	J	2	0	2	l_s+1	M_s: $2l_s$ M_i: $f(i,s)$	
	L	1	0	1	l_s	M_s: $2l_s$ M_i: $f(i,s)$	
	M	2	0	3	$l_s+l'_{s_1}+l'_{s_2}+2$	M_{s_j}: $l_{s_j}+l'_{s_j}+1$ $M_i\in T_{\langle l_{s_j},v_{s_j}\rangle}$: $f(i,s_j)$ M_s: $l_s+f(s,s_j^*)-1$ other M_i: $f(i,s_j^*)$	
	P	$\mathit{min}(\lceil\log p\rceil+1,h)$	0	$\mathit{min}(2p,\lceil\frac{n}{2}\rceil)$	$h\cdot\mathit{min}(2p,\lceil\frac{n}{2}\rceil)$	M_{s_j}: $2l_{s_j}-1$, M_{s_r}: $2l_{s_r}$ M_i: $\mathit{max}(\{f(i,s_j)\mid\forall s_j\})$	

Remarks: S - setup, J - join, L - leave, M - merge, P - partition, Message size and size of saved data are given in $\log q$ bits,

$$f(\alpha,\beta)=\begin{cases} l_\alpha-\lfloor\log|v_\alpha-\lfloor v_\beta/2^{l_\beta-l_\alpha}\rfloor|\rfloor, & \text{if } l_\alpha\leq l_\beta \\ l_\beta-\lfloor\log|v_\beta-\lfloor v_\alpha/2^{l_\alpha-l_\beta}\rfloor|\rfloor, & \text{if } l_\alpha>l_\beta \end{cases}$$

$k\in\{1,\dots,h-1\}$ is the smallest integer such that $\lfloor v_i/2^k\rfloor$ is even.

8. Discussion

With respect to our complexity analysis and introduced notion of verifiable trust relationship we discuss the suitability of the optimized protocols for various mobile ad-hoc groups.

8.1. Homogeneous Groups

Static groups Although μBD does not provide verifiable trust relationship it has low computation costs per member and good communication trade-off. Thus, μBD is sufficient for applications with lower level of security. However, for a higher security level we suggest to use μTGDH that provides cost fairness for homogeneous groups and verifiable trust relationship, and fulfills all security requirements.

Dynamic groups μTGDH is the best choice for homogeneous ad-hoc groups with frequent dynamic events because of the uniform distribution of computation and memory costs due to the balanced tree. Although μTGDH has higher communication costs for setup, join and partition compared to μSTR, it still provides a good trade-off between the message size and the computation costs.

8.2. Heterogeneous Groups

Static groups For static heterogeneous ad-hoc groups we compare μBD, μCLIQUES and μSTR-H setup protocols. μSTR-H has a better communication efficiency than μBD, although μBD requires only 3 multiplications. Hence, we suggest to use μSTR-H if communication constraints are more significant than computation constraints. μSTR-H should also be used to achieve a higher security level, because neither μBD nor μCLIQUES do provide verifiable trust relationship. However, if a lower security level is sufficient and communication is not the primary constraint then either μBD or μCLIQUES may be chosen: if n is not very large and additional costs of $n-1$ multiplications with small integers are negligible then μBD is an optimal solution since its communication complexity is better than that of μCLIQUES; otherwise if additional costs are non-negligible due to large n then μCLIQUES is a better choice and the device with the highest performance ratio should act as the sponsor.

Dynamic groups For dynamic heterogeneous ad-hoc groups we recommend to choose between μSTR-H and μCLIQUES. μSTR-H provides best communication efficiency (especially in case of merge) and a higher level of security (i.e., verifiable trust relationship) than μCLIQUES that requires a constant number of computations for most of the devices. In groups where communication constraints are more significant or security level must be high we suggest to use μSTR-H. In groups where computation constraints are more significant or a lower security level is sufficient μCLIQUES may also be considered as a choice. Due to the highest computation costs of the sponsor in μCLIQUES we suggest that if the major part of the involved devices is less powerful (e.g., mobile phones) and there are only several higher-performance devices (e.g., laptops) then μCLIQUES should be used with the highest-performance device acting as a sponsor. However, if the group is mixed, e.g., the number of involved mobile phones, PDAs and laptops is almost equal then μSTR-H is a better choice, because of the non-uniform distribution of computation costs with respect to the cost fairness requirement.

9. Conclusion

In this paper we have revisited CGKA protocols from the perspective of mobile ad-hoc networks and devices. We have optimized all described protocols to achieve better computation, communication and memory costs, and discussed the suitability of the modified protocols for static and dynamic homogeneous and heterogeneous mobile ad-hoc groups.

Obviously, none of the protocols can be used as a universal solution for all described classes of mobile ad-hoc groups. Moreover, best suitable protocol should be chosen according to our recommendations under consideration of the expected dynamic behaviour in the group, required level of the security for the application, and priority between computation and communication constraints (i.e., a trade-off between device performance and network performance).

References

[1] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik. On the performance of group key agreement protocols. *ACM Trans. Inf. Syst. Secur.*, 7(3):457–488, 2004.

[2] R. Bhaskar. Group key agreement in ad hoc networks. Technical Report RR-4832, INRIA Rocquencourt, May 2003. `http://www.inria.fr/rrrt/rr-4832.html`.

[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] S. Capkun, L. Buttyn, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, 2003.

[5] A. Joux and K. Nguyen. Separating decision diffie-hellman from diffie-hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, 2001. `http://eprint.iacr.org/`.

[6] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2003.

[7] 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.

[8] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, 2004.

[9] J. Luo, P. T. Eugster, and J.-P. Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In *INFOCOM*, 2003.

[10] M. Manulis. Key agreement for heterogeneous mobile ad-hoc groups. In *Proceedings of 11th International Conference on Parallel and Distributed Systems (ICPADS 2005) Volume 2 International Workshop on Security in Networks and Distributed Systems (SNDS 2005)*, pages 290–294. IEEE Computer Society, 2005.

[11] U. M. Maurer and S. Wolf. The Diffie-Hellman protocol. *Designs, Codes and Cryptography*, 19:147–171, 2000.

[12] Standards for Efficient Cryptography Group (SEC). Sec 1: Elliptic curve cryptography, `http://www.sec.org`, September 2000.

[13] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8), 2000.