# A modular fusion of different GKA protocols for various use-cases

Luca Otting
*Chair of IT Security*
*Technical University of Munich*
Garching, Germany
luca.otting@tum.de

***Abstract*—In an increasingly digital world in which data and knowledge are becoming more important while at the same time facing the wish for privacy, encrypted communication is essential. Especially in areas like messaging or wireless sensor networks we, therefore, need a solution to achieve secure group communication. To ensure privacy, integrity, confidentiality and authenticity it is necessary to employ group key agreement (GKA) protocols. Despite the need for such protocols, group key agreement still poses some significant issues and therefore is actively being researched as current protocols have some major problems making them unsuitable for some use cases.**

why?

**Therefore this paper proposes two combinations of protocols to create a new protocol that coverers various important requirements. This is done by first categorizing protocols with notable and useful features regarding the base protocol, the authentication mechanism, authority hierarchy and dynamic or static character. With this categorization and a performance and security evaluation, the paper concludes that there is not a single best protocol to choose. Instead, it is necessary to combine them. This yields two modular protocol fusions that can be adapted to various requirements and therefore offer a solution that is more suitable than current protocols.**

***Index Terms*—group key agreement, key exchange, tree-based key agreement, identity-based key agreement**

## I. Introduction

Secure group communication is becoming more and more important in today's world, because more and more of our data and knowledge is stored digitally. However unfortunately there are still many open questions to be answered leaving users with some significant flaws in current protocols [3], [4], [9]. Because of the relevancy of group key agreement (GKA) protocols, researchers are constantly proposing new protocols to improve on each other.

Although all these protocols are used to agree upon secret group keys they focus on very different issues and aspects. GKA protocols for wireless sensor networks for example try to reduce the memory and processing overhead and try to solve lightweight authentication [5]–[8] whereas others add the concept of a hierarchy to GKA protocols [4], [6]. However, in modern use cases, we often want to have multiple features and not only a for example lightweight authentication while still being stuck with other major flaws. Therefore we have to identify some common requirements and focus on protocols offering special and useful additions. Given the amount of GKA protocols however have to narrow down the protocols that will be included in this paper.

In general, we can differentiate between

- centralised
- decentralised
- contributory

group key agreement protocols [1]. In centralised GKA protocols, we have a central trusted authority that distributes keys over pair-wise secure connections [1]. Because of that, we will not cover centralised GKA protocols as pair-wise key exchanges are already well-researched [19]. In addition, having a central trusted system creates many more problems not only related to GKA protocols. Those for example are high availability and scalability.

Besides centralised mechanisms, we also have decentralised GKA protocols. Here a member is elected to be the group manager who is responsible for storing information about the group and distributing keys [1]. This however poses similar challenges to a centralised approach with the difference that members have more control by dynamically electing a group leader. Because any member can be the group leader this group of protocols is named decentralised. However, we still elect a single leader. As a single point of failure during the group agreement phase is not desirable this paper also excludes decentralised GKA protocols which elect a key distributor [1].

At last, there are contributory GKA mechanisms. The benefit of this type of protocol is that every device contributes to a common group key and we no longer have a single trusted entity [1] and are more resilient to failures. Additionally, it is more secure to have multiple members in control of a shared secret instead of relying on one entity [1]. Unfortunately, those protocols are still actively being researched and have some unresolved issues. Therefore this paper will focus on contributory group key agreement protocols. The evaluation of contributory GKA protocols will help us to understand the differences between each protocol and identify protocols that could be combined resulting in a more sophisticated protocol.

However because there are currently so many different protocols focusing on different features and goals, it is important to take a look at current or future use cases to identify the most important requirements. By doing so we can only focus on suitable protocols that could be combined with others to create a new approach. Because of their special characteristics, the main protocols evaluated are

- TreeKEM [3] proposed by K. Bhargavan, R. Barnes a
  E. Rescorla
- DC-AAGKA [5] by Qi. Zhang, Y. Gan, Qu Zhang,
  Wang and Y. Tan
- AI-GKA [6] by Z. Wan, K. Ren, W. Lou and B. Prene
- ID-GAKA [8] by L. Tian-Fu and C. Meriske
- AB-AKE [4] by M. Gorantla, C. Boyd and J. Gonzale
- (d)PLAKE [9] by H. Seongho, C. Rakyong and
  Kwangjo

Each of those protocols has its advantages and therefore the possibility to contribute to an improved GKA mechanism.

To give an accurate overview and evaluation, the paper will therefore be structured as follows. The next section describes based on which characteristics we will categorise protocols. In section IV the paper will then categorise all previously mentioned protocols and shortly explain them. Then section V follows with the performance analysis and in section VI the security analysis. At last in section VII the paper proposes some possible new GKA protocols created from the fusion of existing protocols. The paper is then concluded in section IIX.

## II. Background

### A. Decisional Diffie-Hellman (DDH) problem

To understand the concept of security we first have to understand the fundamental assumption under which a group key agreement protocol is considered secure. All protocols this paper covers are secure under the Decisional Diffie-Hellman Assumption [2]. An easy-to-understand explanation of this problem is the following: Given a cyclic group G that can be generated with a generator g, we can not find an efficient algorithm to differentiate between the distributions $\langle g^a, g^b, g^{ab}\rangle$ and $\langle g^a, g^b, g^c\rangle$ in which we choose $a$, $b$ and $c$ at random from $[1, |G|]$ [2]. This means that $g^{ab}$ seems to be chosen randomly from an eavesdropper's point of view. Or to formulate it mathematically correct, the function [2]

$$Pr[A(p, g, g^a, g^b, g^{ab}) = "true"] \\ -Pr[A(p, g, g^a, g^b, g^c) = "true"] = \frac{1}{n^\alpha} \tag{1}$$

is close to zero for any algorithm $A$ running in polynomial time. With $p$ being a prime number, $g$ the generator for the group $G$, $\alpha > 0$ fixed and a large enough $n$. [2]
There is also the Computational Diffie-Hellman (CDH) problem which is a stronger assumption than the DDH problem. The CDH not only asks if the above distributions can be differentiated but also for $g^{ab}$ given $g^a$ and $g^b$ [10]. Therefore if the DDH problem can not be solved it is impossible to solve the CDH problem. Because of that, this paper will only cover protocols that are secure under the DDH or CDH assumption.

### B. Key trees

Besides that, it is also essential to know about so-called key trees, as they are especially important in protocols based on "Tree-based Group Diffie-Hellman" (TGDH) [12].

This explanation will focus on binary trees as such trees are used most often.

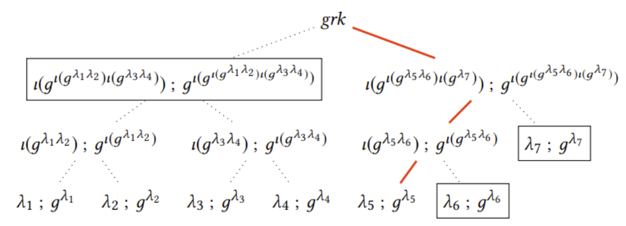
grk
$\iota(g^{\iota(g^{\lambda_1\lambda_2})\iota(g^{\lambda_3\lambda_4})})$ ; $g^{\iota(g^{\iota(g^{\lambda_1\lambda_2})\iota(g^{\lambda_3\lambda_4})})}$
$\iota(g^{\iota(g^{\lambda_5\lambda_6})\iota(g^{\lambda_7})})$ ; $g^{\iota(g^{\iota(g^{\lambda_5\lambda_6})\iota(g^{\lambda_7})})}$
$\iota(g^{\lambda_1\lambda_2})$ ; $g^{\iota(g^{\lambda_1\lambda_2})}$
$\iota(g^{\lambda_3\lambda_4})$ ; $g^{\iota(g^{\lambda_3\lambda_4})}$
$\iota(g^{\lambda_5\lambda_6})$ ; $g^{\iota(g^{\lambda_5\lambda_6})}$
$\lambda_7$ ; $g^{\lambda_7}$
$\lambda_1$ ; $g^{\lambda_1}$
$\lambda_2$ ; $g^{\lambda_2}$
$\lambda_3$ ; $g^{\lambda_3}$
$\lambda_4$ ; $g^{\lambda_4}$
$\lambda_5$ ; $g^{\lambda_5}$
$\lambda_6$ ; $g^{\lambda_6}$

Fig. 1. This figure shows a Diffie-Hellman key tree. $\lambda_i$ denotes to the secret key of a node and the right sight of the colon denotes the public key. [11]

Given a binary tree of height $l$ where the root is on height 0, we have $2^l$ leaves. Those leaves represent the members of a group. Intermediate nodes are roots of subgroups. The root of the whole key tree contains the shared group secret. This structure can also be seen in Figure 1. Keys at intermediate nodes are constructed from their children's key material. [11]. The function used to obtain the key material at these nodes however differs from protocol to protocol. TreeKEM for example uses hash functions to derive secrets at intermediate nodes [3]. It is important to mention that the group secret when using key trees is calculated from bottom to top, which is also visible when looking at Figure 1, as the calculations for the intermediate keys are getting longer with higher nodes. By doing so, all members contribute to a shared group secret. In addition to that, members in groups using key trees only need the public keys of their co-path to verify and calculate a newly generated group secret [12]. The co-path can be obtained by taking the path from a leaf to the root and replacing all nodes with their siblings. Because of that the storage needed is reduced to $\mathcal{O}(\log(n))$. [11]
In the TGDH [12] protocol a Diffie-Hellman key exchange is used to generate the parent node's keys. How this protocol works will be explained in the section about TGDH below to the extent it is needed in this paper. An example of a key tree can also be found when looking at Figure 1.

### C. Burmester and Desmedt protocol

Because the Burmester-Destmedt (BD) [13] protocol is a stepping stone for the protocols covered in this paper it is important to understand the general functionality which makes it easier to follow this paper. In the next section, it will also become clear that the BD protocol is completely different from the TGDH protocol which should be the main takeaway from the explanation of the BD protocol and the following TGDH protocol. The Burmester-Desmedt protocol is a relatively old group key agreement protocol in comparison to the protocols covered in this paper. The protocol needs two rounds for the users to agree on a group secret. The basic principle of this protocol is that at first, the user broadcasts a number computed from a secret random number previously selected by him. Then every user $i$ uses the broadcasted values from $i+1$ and $i-1$ to calculate material later used for the key calculation.

This is again broadcasted. At last, every user takes the last broadcasted values and calculates the shared group secret [13]. More formally, one would define the BD protocol as follows:

1) Given n users $U_1, U_2, \ldots, U_n$ each user $U_i$ chooses a random number $r_i \in Z_q$ and. The user $U_i$ broadcasts $z_i := g^{r_i} mod p$.
2) In round 2 every user $U_i$ calculates $X_i = (z_{i+1}/z_{i-1})^{r_i} mod p$ and broadcasts this value. For user 0 the $z_{i-1} = z_n$ and for user $n$ the $z_{i+1} = z_0$.
3) In round 3 every user $U_i$ finally calculates the key $K_i = (z_{i-1})^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \mod p$.

[13] Unfortunately, there have already been found some other major security flaws making this protocol vulnerable and outdated [14], [15]. Newer protocols often try to improve the BD protocol by adding features or making it resilient against known attacks.

### D. Tree group Diffie-Hellman (TGDH)

The Tree-based group Diffie-Hellman protocol uses as the name suggests a tree structure to organise the group and store key material. It uses the Diffie-Hellman key exchange to calculate a group secret [12]. A simple summarization of this protocol would be the following: TGDH uses key trees where each node is a group member. Every group member has a secret value $\lambda_i$ and a public value $g^{\lambda_i}$ as can also be seen in figure 1. Two leave nodes therefore can use Diffie-Hellman to calculate the private and public key material for their parent node. The secret then being derived from $g^{\lambda_1 \lambda_2}$ with function $l$ and the public key therefore being $g^{l(g^{\lambda_1} \cdot g^{\lambda_2})} = g^{l(g^{\lambda_1 \lambda_2})}$ (figure 1). By repeating this process until the root is reached TGDH calculates a shared group secret [12]. How exactly the mathematics behind Diffie-Hellman works however is not part of this paper and can be found by looking at the TGDH paper [12]. Because of the tree structure and use of Diffie-Hellman TGDH needs to perform $\mathcal{O}(\log(n))$ key exchanges in a group with $n$ members.

## III. Categorization of Group Key Agreement Protocols

There is a multitude of GKA protocols that all focus on different issues and use cases. Therefore the protocols can be and must be categorised by certain traits. Because there have been far too many GKA protocols proposed to include all protocols in this summary it is impossible to focus on every little detail. In addition to that, there are already some older surveys [16], [17] which can be used to learn more about well-researched protocols. Therefore the selection of papers is limited to relatively new proposals with each having at least one special characteristic that makes the protocol suitable for a possible protocol fusion to create a new and improved protocol. The BD [13] and TGDH [12] protocols will not be considered in this paper. However, they have shortly been explained as it is essential to understand that both protocols use completely different mechanisms to agree upon a shared group secret. This is important, as it might be very hard to combine a BD-based protocol with a TGDH-based protocol.

### A. Categorization criterias

Because there are numerous GKA protocols all being slightly different from others one could categorise those protocols by a multitude of attributes. This paper however focuses on four key differences which are important to consider when improving upon existing protocols. Firstly GKA protocols can be of static or dynamic nature. In addition to that one can differentiate protocols by looking at their authority hierarchy or concept of privileges. The third point of interest is the authentication mechanism and lastly this paper evaluates the basis protocol of each mentioned protocol. The basis protocols in this paper can be the BD or TGDH protocol.

This paper will evaluate each of these traits per protocol in the following sub-sections. The categorisation will be important when comparing protocols and drawing conclusions as some features might be hard to combine if the base mechanisms are wildly different from each other or if one protocol does only support static groups while the other supports the addition and removal of users.

*1) Static and dynamic protocols:* Firstly we found that all GKA protocols can be divided into static or dynamic protocols. Dynamic protocols are designed with a changing membership base in mind. Those protocols support adding and removing members. Static protocols however rely on reinitiating the group to perform such operations. This must not be confused with the properties of a certain device or device group. Wireless sensor networks consisting of multiple sensor nodes for instance can be highly dynamic concerning their uptime, reliability and location of deployment. Despite that sensor nodes can still participate in static group key protocols [5], [8] as they are still able to store static group information. If there is for example a group in which only one user is added or removed per day it could make a lot of sense to reinitiate a group as this process is neglectable for a whole day. good

Nevertheless, it is still important to look at whether a protocol is static or dynamic as this can have a huge impact on the usability of a protocol and it might be harder to transfer certain features from a static protocol to a dynamic protocol while still maintaining the same security properties.

*2) Authority hierarchy:* Besides that GKA protocols can also be grouped based on their authority hierarchy. Flat protocols do not provide any levels of different permissions and every member has the same level of authority as all devices are considered to be equal. GKA protocols with an authority hierarchy, however, provide different levels of authority and one sub-group of members can have more permissions than another sub-group. This property for example can be advantageous if you wish to have a group administrator or moderators while still using a contributory GKA protocol.

Without a hierarchy, groups would have to rely on a central authority to block certain operations performed by less privileged members if a user wishes to have a hierarchy. As this feature however is needed in various use cases it is vital to include the concept of privileges in GKA protocols. In addition to that it can be hard to combine a protocol that relies on a group initiator to perform certain actions with a flat protocol.

It is important that this paper only looks at the permissions a user has within a group. A protocol using a trusted third party for example to provide authentication is still considered a flat protocol if all users can initiate all operations without restrictions. The trusted third party is not considered to be a more privileged user.

Some protocols do not support different member privileges but are still designed with the concept of a hierarchy in mind [4]. In the attribute-based protocol, we are going to talk about later, we have a flat authority hierarchy within a group but can easily achieve a hierarchy between groups.

*3) Authentication mechanism:* A third difference GKA protocols have is the mechanisms they use to achieve authentication. Some protocols use identity-based encryption mechanisms while others rely on an out-of-band authentication mechanism which is not trivial to achieve as you need a secure out-of-band channel to prevent tampering with messages sent over the authentication channel. In some cases, a PKI (public key infrastructure) is used to authenticate members by using certificates and signatures. Identity-based encryption on the other hand include identity information in the key generation process. By doing so a member confirms another member's identity by extracting information from the exchanged key material. In identity-based encryption, the public key can be derived from a user's identity. Using identity-based authentication, therefore, reduces message and computation overhead as no further certificate validations or message exchanges have to be performed. [5]–[8] If Bob for example sends his public key to Alice and previously obtained the public key from a third party that used Bob's mail address "bob@mail.de" for the key generation process, Alice can verify that the public key really is Bob's key. She can do this because the public key is derived from Bob's identity "bob@mail.de", which is known to Alice. Therefore identity-based authentication is often used where lightweight authentication mechanisms are beneficial or necessary. In Identity-based encryption, it is necessary to have a trusted entity responsible for the key generation process which could potentially be a security risk as this trusted entity is responsible for the security of the whole system. The same can however also be said for a centralised PKI. No matter which authentication mechanism is used it is important to note that nearly all GKA protocols rely on a trusted third party to achieve authenticity [4]–[8]. How the user then authenticates with this third party will however not be covered in this paper.

*4) Basis protocol:* At last, one can characterise GKA protocols based on the basis protocol. In general, one can divide all protocols into three categories. There are tree-based GKA protocols stemming from TGDH (Tree group Diffie-Hellman), BD-based protocols and GKA protocols based on neither TGDH nor BM. [6] Even though there are tree-based protocols not performing any Diffie-Hellman key exchange they are rooted in the TGDH protocol. BD-based protocols nowadays are using very different calculations but follow the same principle as the original BD protocol. The last group is relatively small as TGDH and BM protocols have proven to be efficient and secure. Therefore this paper will not focus on this third group.
It is essential to characterise protocols based on the operating basis as the fusion of protocols based on completely different mechanisms is not trivial.

## IV. Protocol Categorisation

### A. TreeKEM

The first protocol this paper will evaluate is called TreeKEM [3]. As the name suggests this GKA protocol uses a key tree to organise the secrets it needs to transmit messages. In contrast to older GKA protocols however, TreeKEM does not have contributory secrets at intermediate nodes. Instead, if a device updates its key a hash of the member's public key is stored at the parent's node. This node's parent then again stores a hash of the key which is now the member's public key hashed twice. Therefore the path from a member that updated the key to the root is always controlled by this user. The protocol as a whole however is still contributory because the new group secret includes the old group secret in the calculations. Therefore all members are contributing to a shared group key. This reduces overhead and improves the performance of the GKA protocol as members only have to evaluate hash functions on the way to the root node. In addition to that, a member only needs to store $\mathcal{O}(\log(n))$ public keys to verify and execute the TreeKEM protocol. Because hash functions are usually very well optimized and often accelerated with hardware this protocol is perfect for very large groups.

Besides that, it is important to note that TreeKEM does not only restrict keypairs used to Diffie-Hellman key pairs but allows any keypairs supporting key encapsulation. This for example allows the use of identity-based keypairs. The purpose of identity-based keypairs is to include identifying information in the key material which enables receivers of a member's public key to verify the authenticity of a member. By doing so TreeKEM's authentication would function in a similar way to for example the ID-AGKA [18] protocol. Without the use of identity-based keys TreeKEM relies on a PKI to authenticate group members. Which mechanism is used is therefore up to the user of the TreeKEM protocol.

not explaine

This protocol has been designed with a dynamic membership base in mind and therefore supports removing and adding users to the group. Because of that, it can be categorised as a dynamic GKA protocol.

One notable feature of this protocol however is that TreeKEM can be executed asynchronously, which means that members do not have to negotiate with other members before certain group operations can be processed. Every member has a local state that eventually will converge to the current global state after processing all group operations. Asynchronous operation is often a desirable feature when communicating with group members that are not highly available and hard to synchronize. One example of this would be a chat platform like WhatsApp.

There are however some issues that can occur because operations can be executed asynchronously. For example, we have to decide what happens if two users remove each other at the same time or add a new node at the same location in the key tree. This asynchronicity can also impact the security of TreeKEM if for example two key updates are executed at the same time. The security analysis however will follow in a section below. The TreeKEM paper proposed several options to solve these problems. One of them is the use of a central authority that enforces an absolute order the other one negotiating an absolute order among the group members. Secondly, there is also no concept of authority hierarchy in TreeKEM. Every member has the same level of authority and can remove or add members.

!

## B. DC-AAGKA

The second GKA protocol is called DC-AAGKA [5]. In comparison to TreeKEM, this GKA protocol approaches the key agreement very differently and focuses on other issues. The main goal of this protocol is to enable group key agreement between members in different domains. This means that we can have an enclosed ecosystem of sensor nodes and a certification authority (CA) and still, be able to contribute to a key for communicating with other members in other domains.

If for example, three different companies are designing a new product they can individually exchange messages within their own company (domain) but also agree upon a shared secret to communicate changes securely to the other companies.

However, for this to be possible all certification authorities must exchange a common private and public key pair in a non-trivial way so that group members can be authenticated by members in other domains.

Because every domain must have a centralised CA this protocol, unfortunately, comes with some issues like trust in a single entity and the possibility of a single point of failure. On the other hand, we still have the benefit of a CA in each domain. Therefore it would be possible to use a GKA protocol to decide upon a key pair for the CAs. This would allow every CA to contribute to the shared secret between them.

yes

This protocol can also be categorised as a dynamic GKA protocol, as it supports members joining or leaving the group. This makes the protocol suitable for long-lived groups in a dynamic environment. In addition, this protocol is based on the BM GKA protocol and needs a constant number of rounds to establish a group session, which makes it suitable for large groups and low-power devices. Besides that, it has to be said that DC-AAGKA is a flat GKA protocol and has no concept of different authority levels. Every device can add or remove group members.

## C. AI-GKA

Another protocol is the AI-GKA [6] protocol. The important difference of it is the focus on privacy protection while still providing user authentication and using identity-based encryption. Other than protecting privacy this protocol also is suited for a dynamic group as members can be added and removed. Like the previous protocol, AI-GKA is also based on the BM GKA protocol and only needs a constant number of rounds.

Because this protocol tries to preserve privacy and does not require too much computation power and storage it is well-suited for consumer devices like smartphones or laptops.

Privacy protection is achieved by the group initiator giving every member a pseudonym which is then used in the group key agreement and message exchange. Because of this pseudonym, outsiders can't know who is part of a certain group. Only group members know this private information. Therefore AI-GKA can still use identity-based encryption while maintaining privacy from an outsider's perspective.

how dynamic

However, as in all identity-based GKA protocols, AI-GKA relies on a trusted server to generate private keys based on a user's real identity. This real identity is not a secret, it is just replaced by pseudonyms to hide which users are members of a group. An adversary can obtain all pseudonyms that are part of a group but can not connect them to any real identities.

Other than that in this protocol, there are also two different authority levels. First, there is the group initiator and secondly all other members. The paper for AI-GKA proposes a mechanism for creating a group, adding and removing members that rely on the group initiator to perform some necessary steps in the protocol's execution. Because of that, the group initiator has control over the membership base and not everyone can invite new members or remove current members. This makes the group initiator the administrator of the group. Even though this protocol did not try to solve the problem of different privileges it is a nice side effect of the protocol. Overall AI-GKA is still contributory even though it has a group initiator as all members still have to contribute to the group secret.

## D. ID-GAKA

A fourth relatively new protocol is the ID-GAKA [8] or its extension the ID-GAKA-PFS [8] protocol. The difference here is that the latter also provides perfect forward secrecy by adding some additional logic to the protocol execution. Because perfect forward secrecy is an important security trait, this paper will only focus on this variant in the following analysis. Therefore ID-GAKA and ID-GAKA-PFS will be used interchangeably.

ID-GAKA again is an identity-based GKA protocol and therefore relies on a central entity to generate a key pair based on a user's identity. The actual group key agreement makes use of chaotic maps to agree upon a shared secret. By doing so the protocol only needs to execute 8 chaotic maps and 5 hashes to establish a group secret. Users do not have to perform expensive modular exponential operations or symmetric en/decryption operations.

In addition to that this protocol can be classified as static as it does not support adding or removing group members. The paper proposes to set a time window of for example 30 minutes before reinitializing the group to update the group key. Because of that however group members do not have to store any long-term data besides the group secret. This reduction in

whats a chaotic map?

long-term storage cost as well as the increased computational performance makes this protocol work very well in groups with low power and low storage devices.

Unfortunately when a new user is added or removed the protocol has to be executed again which is equal to the initialization of a completely new group. Because this ID-GAKA however is very lightweight this comes with less overhead than other GKA protocols would produce.

By not relying on TGDH this protocol belongs to the BD-based protocols. The reason for that is the same principle of computing a number based on a selected random number and then broadcasting it to all group members. The group members next to the broadcasting member can then use this value to calculate material for the group secret. After broadcasting this intermediate key material every user can derive the group key. ID-GAKA(-PFS) also can be considered a flat protocol as it does not differentiate between users with different privileges.

The main goal of this protocol is to provide a lightweight GKA protocol that can be executed on low-power and low-storage devices.

### E. AB-AKE

Another very interesting protocol is the attribute-based authenticated key exchange protocol AB-AKE [4] even though the paper does not propose join and leave operations making the protocol a static group key agreement protocol.

The important feature however is the use of access policies. In AB-AKE a user can specify an access policy which gives only authorised group members the ability to derive and contribute to the group key. In this protocol, every user has a set of attributes. If those attributes match the access policy the user can calculate the shared group secret. This attribute-based encryption is an extension of identity-based encryption and works similarly. The main difference is that instead of a user's identity the user's attributes are used to authenticate the user.

This makes it possible to establish a group with for example all temperature sensors without having to know the exact identity of each sensor. While this GKA protocol has a flat hierarchy within the group it can be used in scenarios where members of for example a forum have different privileges. By setting an access policy to only allow administrators for example we can establish secure communication between all administrators. By executing the AB-AKE protocol with different access policies we can't achieve a group with different privileges internally but have different groups with different privileges. This makes the protocol suitable for situations in which a user does not know the identity of his communication partners but knows their attributes. is that id?

The AB-AKE paper proposes a one-round key exchange protocol for two parties but states that it can easily be extended to a group setting only needing three rounds. This can be done by combining AB-AKE with the BD protocol making it a BD-based GKA protocol. In the next section, AB-AKE will refer to this BD-based GKA protocol. One drawback of this protocol however is that it does not have forward secrecy and therefore is vulnerable after a group member has been compromised.

### F. (d)PLAKE

Lastly, this paper covers the (d)PLAKE [9] protocol which is based on TGDH and supports adding and removing group members. Unlike TreeKEM this protocol uses a blockchain to achieve authenticity. This eliminates a single central entity that authenticates users. The PLAKE protocol uses a proof of work whereas dPLAKE uses a delegated proof of stake. How exactly those two proofs and blockchains work, however, is not part of this paper. All that is important is that by using those mechanisms a user who wants to join a new group for example has to perform work to join a group to show that he is willing to invest resources to join the group. By doing so, it is harder for attackers to generate false blocks and join a group. When using the delegated proof of stake the user has to authenticate himself with a group member. The paper for dPLAKE however does unfortunately not propose such a one-way authentication function. Other than that, the protocol also achieves an absolute order of group operations as a blockchain is a chain of strictly ordered elements. This would, for example, eliminate the problem of two asynchronous remove operations or two asynchronous key update operations that lead to issues in TreeKEM.

This protocol on the other hand also brings some significant drawbacks. If an adversary for example has enough resources to solve the proof of work in much less time than expected the blockchain approach with proof of work would not ensure authenticity for a joining member. In addition to that it usually also takes multiple minutes to hours to join a group depending on how hard the puzzles for the proof of work are. This makes it unsuitable for a highly dynamic group with a fast-changing membership. In addition to that, the dPLAKE protocol does not specify the one-way authentication function needed for the protocol's execution when using a delegated proof of stake. Therefore this protocol still needs some more research before it can be used in a real-world setting.

## V. Performance analysis

### A. Number of rounds

The first metric we have to consider when talking about performance is the number of rounds required for a group to agree upon a shared secret. As most GKA protocols try to achieve scalability GKA protocols must have a constant number of rounds. TreeKEM, DC-AAGKA, ID-GAKA(-PFS) all require only two rounds to generate a group key [3], [5], [8]. AI-GKA and AB-AKE need one more round resulting in 3 rounds [4], [6]. Therefore all these protocols have a constant number of rounds.

(d)PLAKE on the other hand uses the standard TGDH protocol in addition to its blockchain authentication. While TGDH itself requires only 2 rounds [12] to add a new member to a group it requires $\mathcal{O}(\log(n))$ rounds to establish a new group. The reason for that is the functioning of the key tree.

| | Protocols | | | | | |
|---|---|---|---|---|---|---|
| **Name** | *TreeKEM* | *DC-AAGKA* | *AI-GKA* | *ID-GAKA* | *AB-AKE* | *(d)PLAKE* |
| **Base protocol** | TGDH | BD | BD | BD | BD | TGDH |
| **Auth** | ID/PKI | ID | ID | ID | AB | BC |
| **Priviledges** | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| **Static vs dynamic** | d | d | d | s | s | d |
| **# of rounds** | 2 | 2 | 3 | 2 | 3 | $\mathcal{O}(\log(n))$ |
| **PCS** | ❍ | ✓ | ✓ | - | - | ✓ |
| **FS** | ✓ | ✓ | ✓ | - | - | ✓ |

Fig. 2. ID = identity-based encryption, PKI = public key infrastructure, AB = attribute-based encryption, BC = blockchain, PCS = post-compromise security, FS = forward secrecy, d = dynamic, s = static

As earlier described this tree is built from the bottom up. Therefore at first two leave nodes need to exchange their public key to calculate the key pair for the intermediate node. Then in the next round, we can use the public keys of those intermediate nodes to calculate the key pair of the next intermediate node. By doing so we need $\mathcal{O}(\log(n))$ rounds to establish a new group.

Even though TreeKEM also uses a key tree this is not the case for TreeKEM as their intermediate nodes are controlled by a single member of the group [3]. However, it has to be noted that the base TGDH protocol in (d)PLAKE could be replaced by TreeKEM as the public ledger mechanism only adds another step to the protocol execution but does not interfere with the TGDH protocol [9]. In theory, it could therefore be achieved to have a (d)PLAKE protocol with a constant number of rounds.

## B. Computation overhead

It is however hard to draw a conclusion from only looking at the number of rounds as there are more factors impacting a protocol's performance. Because the creation of a group is often the most expensive operation this evaluation will mostly focus on creating new groups [3], [5]–[8]. If there is however a more expensive operation we will consider it. Besides that, considering join or leave operations would make it hard to compare static and dynamic protocols.

In TreeKEM the initiator of a group operation always has to do more work than the other group members. A group creation requires $\mathcal{O}(n)$ encryptions for the initiator and $\mathcal{O}(\log(n))$ hashes and one decryption operation for the other members [3]. The overall cost to create a new group therefore overall lies in $\mathcal{O}(n)$. For the non-initiator users the calculation overhead however is significantly lower [3].

DC-AAGKA on the other hand needs $2n$ bilinear pairings and $3n + 2$ scalar multiplications to establish a group session [5]. What exactly these operations do is not important for the performance analysis but we need to know that those pairings are relatively expensive in comparison to multiplications or exponentiations [3].

ID-GAKA-PFS needs to calculate 8 Chebyshev chaotic maps and 5 hashes per member to establish a group session. This however just shows the computationally expensive operations. As ID-GAKA-PFS is still contributory each user still has to execute $\mathcal{O}(n)$ relatively lightweight operations like multiplications or bitwise operations. Because these operations however are relatively cheap compared to the expensive operations mentioned above we can say that we only need a constant number of those complex operations. [8].

AI-GKA again needs $\mathcal{O}(n)$ computations on a single member to establish a group session as the group initiator has to send encrypted group creation messages to every member using individual keys. These messages are needed to transmit the pseudonyms while still not having agreed upon a key. This leads to $n$ encryption/decryption operations. Other than that overall $2n$ bilinear pairings, $n$ signature verifications and $n$ point multiplications have to be calculated [6].

For AB-AKE we again need a linear number of calculations per user. Unfortunately, the paper does not specify the exact number of calculations needed to establish a group session. Because this protocol however adds the feature of attributes and access policies and is based on Burmester-Destmedt [4] it is as expensive or even more expensive as other protocols based on the BD protocol that focus not on performance [5], [6].

Lastly, there are dPLAKE and PLAKE. PLAKE relies on proof of work and therefore requires some heavy computations. The paper proposes a proof of work that takes about three hours for a joining member to complete. dPLAKE with its proof of stake however is much faster. [9] Because both protocols rely on the base TGDH protocol, PLAKE and dPLAKE require more expensive operations than the TreeKEM protocol. The reason for this is the use of Diffie-Hellman key exchanges to compute the keys for a parent node. This is a very expensive operation as it requires a lot of multiplications and exponentiations [12]. Therefore even dPLAKE which uses delegated proof of stake is still more expensive than TreeKEM.

With all this said we now can conclude. As we can see every protocol has linear complexity when creating a new group as every member contributes to the group key. It is very clear that the PLAKE and dPLAKE algorithms do not focus on performance but on other aspects of GKA protocols. Therefore they are the worst-performing protocols and are not suitable for highly dynamic groups.

The by far fastest and most lightweight protocol is the ID-GAKA-PFS. This however is no surprise as it focuses on a lightweight implementation for wireless sensor nodes with very little computation and storage resources.

All other protocols except TreeKEM perform very similarly by simultaneously focusing on different issues regarding GKA protocols.

TreeKEM also needs to perform $\mathcal{O}(n)$ operations for the group initiator however join and remove operations are much faster than in other protocols. Because the binary tree-key members always only have to perform one decryption and $\mathcal{O}(\log(n))$ hashes [3]. Based on the hash algorithm those hashes can be calculated very fast. Even with one million group members, we would only have to calculate 19 hashes. Calculating 19 hashes can be faster than for example calculating several bilinear pairings or chaotic maps as used in other protocols. This is especially the case in modern devices that often include hardware acceleration for hashing algorithms. In addition, TreeKEM can add or remove users by not sending more than two messages.

This makes it very suitable for highly dynamic large groups [3]. With all this in mind, we can say that TreeKEM is most suitable for very large dynamic groups and ID-GAKA-PFS should be used if we only need a very lightweight GKA protocol. When only looking at the performance (d)PLAKE is the worst-performing protocol. However, we can not describe one protocol as better than another by just looking at the performance as (d)PLAKE for example focuses on a different aspect in GKA. This comparison however is still necessary to understand why it would for example make sense to combine TreeKEM and (d)PLAKE to make (d)PLAKE a viable option.

### C. *Storage cost*

The lowest long-term storage costs are achieved by using static GKA protocols (e.g. ID-GAKA [8] or AB-AKE [4]) as they do not need to store long-term keys or information for any calculations after the group has been established. Static GKA protocols only have to store the static group key to participate in the group. Dynamic groups however have to store the long-term values to handle user add or remove operations (e.g. TreeKEM [3]).

In those scenarios, the tree-based protocols perform especially well as they only need to store $\mathcal{O}(\log(n))$ secrets [3], [12].

Some dynamic protocols [6] have to store $\mathcal{O}(n)$ secrets to compute a new group key after a user has been added or removed.

Because (d)PLAKE additionally requires a public ledger these two variants need additional storage space resulting in $\mathcal{O}(h)$ storage costs [9] with $h$ being the number of group operations. However, there are also dynamic protocols [5] that do not need to store the individual key contributions of each member but simply use the previous group key in combination with a new contribution as a new shared secret.

One could now think that a dynamic protocol that only needs constant storage space would be the best choice for devices with only a low amount of storage space. This however is not the case with the dynamic protocols based on Burmester-Destmedt that only need constant storage for long-term values. These protocols still need to store $\mathcal{O}(n)$ values during the group initiation as every member contributes to the shared group secret. Therefore these protocols still need a considerable amount of short-term storage space.

Because of that TreeKEM is the best choice for devices with low storage space as it only needs to store $\mathcal{O}(\log(n))$ secrets. If there is enough storage space for the initiation phase then DC-GAKA or a static protocol like ID-GAKA-PFS or AB-AKE are also great choices.

## VI. Security

The most important feature of a GKA protocol is security as having secure communication is the whole purpose of GKA protocols. Security however is a broad topic and can have many interpretations. Therefore this paper will focus on confidentiality, integrity and authenticity as well as on post-compromise and forward secrecy as these are essential attributes, especially for highly dynamic groups. To provide a thorough evaluation we first have to explain these concepts.

### A. *Definitions*

*1) Confidentiality:* A message is considered confidential if a passive eavesdropper can't read the unencrypted message transmitted to a destination. Only the sender and the intended receiver should be able to decrypt and read a message. Post-compromise security and forward secrecy are part of this concept. **Post compromise security (PCS)** guarantees that users can not calculate a shared group secret if they have been removed from the group or that an attacker can no longer read messages after the compromised device updated its key. **Forward secrecy (FS)** on the other hand ensures that newly added members are not able to compute previous group keys and therefore can not decrypt messages sent before the join operation [3].

*2) Integrity:* Integrity means that a message is still the original message when the receiver gets it and has not been changed either by an active adversary or transmission errors [3].

*3) Authenticity:* A message is considered authentic if it really comes from the sender the receiver thinks it comes from. If a message is authentic the sender has to be authenticated which means that the receiver checked the identity of the sender and is sure that the sender is who he claims to be [20].

### B. *Evaluation*

With all these attributes now explained we can compare the different protocols. Most protocols guarantee relatively strong security standards whereas others are lacking some. In general, all protocols covered in this paper provide integrity and authenticity as well as basic confidentiality. However, some protocols ensure stronger security guarantees than others and for example, offer perfect post-compromise security or forward secrecy.

As integrity can be achieved by for example using a signature this feature can be achieved with every protocol and is therefore not discussed in more detail.

In addition to that all protocols either use identity-based encryption or a PKI to authenticate users. Using signatures ensures that all protocols provide authenticity. Because of that, we will only talk about confidentiality, in the following evaluation.

TreeKEM can offer strong forward secrecy and post-compromise security [3]. This is especially desirable for highly dynamic groups as users should only be able to read messages they received while being group members. However, for concurrent group operations, which are supported by TreeKEM an additional key update is needed to achieve post-compromise security. In addition to that TreeKEM does not provide PCS if concurrent key updates are allowed as the new group keys are then delivered using the old keys. [3] Therefore it is necessary to achieve a strict order with for example a blockchain or central authority if the user wants perfect PCS.

DC-AAGKA also provides post-compromise and forward secrecy which also makes DC-AAGKA suitable for highly dynamic groups. [5] However as DC-AAGKA uses identity-based encryption it relies on a private key generator that at some point has access to the user's private key. The reason for that is that the private key generator is used to calculate a private key that can decrypt messages that have been encrypted with a public key derived from a user's identity. To mitigate this single point of failure the private key generation could be distributed among multiple servers. With this, every server would only own a small fragment of a user's private key.

AI-GKA provides confidentiality and integrity. Even though the AI-GKA paper does not mention integrity it achieves this feature during the group key agreement as all members verify the calculated key by sending it to the group initiator which then checks the key. This also ensures additional security as all members can verify if they received all contributions for the shared group secret. If a message would have been corrupted at least one member would have sent the wrong group key to the initiator indicating an error. In addition to that AI-GKA also provides forward and post-compromise secrecy as a new key is calculated upon joining or leaving the group. The new keys can not be derived from previous keys without being a group member. [3]

ID-GAKA again offers confidentiality and authenticity as this protocol is based on identity-based encryption which makes it easy to verify the authenticity of a message. During the group key agreement phase, ID-GAKA also provides message integrity as all group members perform a verification before calculating the key that ensures that no message has been corrupted. Because this protocol is a static GKA protocol and a new session has to be established to add or remove new group members this protocol also provides forward and post-compromise secrecy when using newly generated random values. The knowledge of a key from a previous session does not allow an attacker to decrypt messages from a completely new group.

AB-AKE provides confidentiality and authenticity as it again uses identity-based keys to encrypt messages. The base version of AB-AKE does not provide forward secrecy, however the paper states that in combination with the BD-protocol AB-AKE can achieve forward secrecy [4].

(d)PLAKE does offer PCS and FS as it is based on TGDH which also guarantees these features. Besides that, we could also achieve integrity as mentioned above. Unfortunately, there is still a problem when looking at the authenticity aspect. If an attack for example would have access to a much faster computer the puzzles provided to the joining member could be solved much faster than expected. This would allow attackers from joining the group without showing the willingness of investing time [9]. For the delegated proof of stake, we have another problem as the paper for (d)PLAKE does not specify a one-way authentication function to authenticate with the delegate [9]. Until these problems are not solved there is still no guarantee of authenticity. [9]

## VII. Discussion

Because there are nearly unlimited areas in which GKA protocols are or could be used this paper chose the examples of chat platforms and sensor networks. These examples will be used to explain why certain protocols have been chosen to be included in a protocol fusion. Despite that, these newly created combinations can still be used in different use cases as the three examples cover a relatively broad set of requirements.

### A. 1st proposal

Our first proposal is the fusion of TreeKem and the privacy-preserving AI-GKA protocol. The reason for that is that people nowadays demand more privacy protections. Especially because platforms like WhatsApp can only collect data about you that for example tells the company in which chats you are and with whom you are chatting, as Chats are end-to-end encrypted and therefore don't allow companies like Meta to process your messages. Because groups on chat platforms however can be quite large, smartphones have limited storage space and can be very dynamic, in regards to the membership base, TreeKem contributes to fulfilling these important requirements of chat platforms. As TreeKem supports all key pairs that support key encapsulation [3] and the base principal of AI-GKA is the mapping of pseudonyms to real identities to use identity-based encryption, AI-GKA can easily be integrated with TreeKem.

This combination of two protocols can be extended with the cross-domain DC-AGKA protocol to allow communications with members across different chat platforms. One big issue many people currently face is that they can not move away from WhatsApp as they are bound to it if they want to continue to be included in social activities. With the possibility to create groups across domains, people could freely decide which chat platform they would like to use. With this groups between Signal and WhatsApp members would for example be possible if both companies would adopt this protocol. As the cross-domain authentication is achieved through CAs in each domain [5] that can be implemented with a separate GKA mechanism it could also be easily integrated with TreeKem despite being a BD-based protocol.

We can also add (d)Plake to this combination in addition to identity-based authentication or to simply achieve a strict ordering of group operations which is required by TreeKem to achieve post-compromise security [3]. Other than that, the group history can be documented with a public ledger [9] if it is important to know at which point users were part of a group. This could for example be the case for military messaging platforms. Besides that with the combination of TreeKEM and (d)Plake we could also use the concept of (d)Plake while still having acceptable performance.

On such messaging platforms it would for example also be great to be able to establish a group session with all officers of a certain rank without having to know their identity. A way to achieve this would be to add the attribute-based GKA protocol AB-AKE to our protocol combination. This however would require some further research as the AB-AKE mechanism is not trivial to add when using a tree-based protocol like TreeKEM or (d)Plake. The fact that TreeKEM however allows any key pairs supporting key encapsulation [3] would make it easier to add AB-AKE.

It is also interesting that you can add more protocols based on your needs and requirements. Therefore you can either combine TreeKEM, AI-GKA, DC-AGKA, (d)Plake and AB-AKE or only TreeKEM and AI-GKA or TreeKem and (d)Plake, based on your needs.

The important feature of this protocol fusion is that it is highly modular and with some work easy to adapt to different use cases.

### B. 2nd proposal

The second proposal of this paper is the fusion of the lightweight ID-GAKA protocol with the attribute-based GKA protocol AB-AKE. This protocol would then be perfectly suited for use in wireless sensor networks [8]. On the one hand, it is performant and lightweight while at the same time for example initializing a group with all gateways or for example all temperature sensors without having to know every single sensor node. This however would be quite complex to achieve and would require more research as it is not trivial to combine the AB-AKE protocol that only allows users to derive the key that fulfils a certain policy [4] with the lightweight static ID-GAKA protocol relying on chaotic maps [8]. However, with some research, it might be possible to apply the concept of using chaotic maps for AB-AKE. If this protocol however is possible then it would make communication in wireless sensor networks much easier. To then also establish group sessions between sensor nodes in different domains, the cross-domain protocol could be added. This would solve for example the problem to achieve secure communication between IoT devices from different manufacturers. If your light sensors for example are in a separate domain as your blinds it would be quite easy to control your blinds by the light sensor input when using the DC-GAKA protocol.

### C. Summary

With the implementation of the two proposals we could achieve versatile protocols that can be useful for various applications. It would not only be a great improvement in science or certain niches, but also for everyday people using chat platforms or IoT devices.

However there are of course still some problems that need to be solved. For example, the integration of the AB-AKE protocol in other BM protocols or even TGDH-based protocols like TreeKEM.

## VIII. Conclusion

Even though the evaluation of performance, security and usability of GKA protocols did not yield a single protocol that can be generally considered better than others it helped to identify which protocols are suitable to combine. With the combinations proposed in this paper, we can achieve a protocol that is suitable for a wide variety of use cases. However, to implement them in real-world scenarios there is still much to be researched in the field of GKA protocols.

When however combining protocols we can achieve a more secure and versatile protocol that can be replaced the current GKA protocols or be implemented in applications where we currently rely on a two-party key exchange.

## References

[1] Y. Amir, Y. KIM, C Nita-Rotaru and G. Tsudik, “On the Performance of Group Key Agreement Protocols” ACM Transactions on Information and System Security, Vol. 7, No. 3, August 2004, Pages 457-488.

[2] D. Boneh “The Decision Diffie-Hellman Problem“ Computer Science Department, Stanford University, Stanford, CA 94305-9045

[3] K. Bhargavan, R. Barnes and E. Rescorla, “TreeKEM: Asynchronous Decentralized Key Management for Large Dynamic Groups“

[4] M. Gorantla, C. Boyd and J. Nieto “Attribute-Based Authenticated Key Exchange“, ACISP 2010, LNCS 6168, pp. 300-317, 2010. Springer-Verlag Berlin Heidelberg 2010

[5] Z. Qikun G. Yong, Z. Quanxin, W. Ruifang and T. Yu-an “A Dynamic and Cross-Domain Authentication Asymmetric Group Key Agreement in Telemedicin Apllication“

[6] Z. Wan, K. Ren, W. Lou and B. Preneel, “Anonymous ID-based Group Key Agreement for Wireless Networks“

[7] S. Shin and T. Kwon, “A Privacy-Preserving Authentication, Authorization and Key Agreement Scheme for Wireless Sensor Networks in 5G-Integrated Internet of Things“, Digital Object Identifier 10.1109/ACCESS.2020.2985719

[8] T- Lee and M. Chen, “Lightweight Identity-Based Group Key Agreements Using Extended Chaotic Maps for Wireless Sensor Networks“, IEEE SENSORS JOURNAL, VOL. 19, NO. 22, NOVEMBER 15, 2019

[9] S. Han, R. Choi and K. Kim, “Adding Authenticity into Tree-based Group Key Agreement by Public Ledger“, Copyright 2019 The Institute of Electronics, Information and Communication Engineers

[10] F. Bao, R. Deng and H. Zhu, “Variations of Diffie-Hellman Problem“, ICICS 2003, LNCS 2836, pp. 301-312, 2003. Springer-Verlag Berlin Heidelberg 2003

[11] K. Cohn-Gordon, C. Cremers and L. Garratt, “On Ends-to-Ends Encryption“, CCS ’18, October 15-19, 2018, Toronto, ON, Canada, 2018 Copyright held by the owner/author(s), ACM ISBN 978-1-4503-5693-0/18/10.

[12] Y. Kim, A. Perrig and G. Tsudik, “Tree-Based Group Key Agreement“, ACM Transactions on Information and System Security, Vol. 7, No. 1, February 2004, Pages 60-96.

[13] M. Burmester and Y.Desmedt, “A secure and scalable Group Key Exchange system“, Computer Science Department, Florida State University, Tallahassee, FL 32306-4530, USA

[14] D. Sun and Y. Tian, “Member Tmapering Attack on Burmester-Desmedt Group Key Exchange Protocol and Its Countermeasures“

[15] M. Baouch, J. Lopez-Ramos, R. Schnyder and B. Torrecillas, “An active attack on a distributed Group Key Exchange system“, November 10, 2021

[16] Z. Mohammad, A. Abusukhon and T. Qattam, “A Survey of Authenticated Key Agreement Protocols for Securing IoT“, 978-1-5386-7942-5/19/$31.00 ©2019 IEEE

[17] S. Rafaeli and D. Hutchison, “A Survey of Key Management for Secure Group Communication“, ACM, Inc., 1515 Broadway, New York, NY 10036 USA

[18] K.C. Reddy and D. Nalla, “Identity Based Authenciated Group Key Agreement Protocol“, A. Menezes, P. Sarkar (Eds.): INDOCRYPT 2002, LNCS 2551, pp. 215-233, 2002

[19] Nan Li, ”Research on Diffie-Hellman key exchange protocol,” 2010 2nd International Conference on Computer Engineering and Technology, Chengdu, China, 2010, pp. V4-634-V4-637, doi: 10.1109/ICCET.2010.5485276.

[20] Varga, Somogy and Charles Guignon, ”Authenticity”, The Stanford Encyclopedia of Philosophy (Spring 2020 Edition), Edward N. Zalta (ed.), https://plato.stanford.edu/archives/spr2020/entries/authenticity/