Discover Data

Research

Network impact analysis on the performance of Secure Group Communication schemes with focus on IoT

Thomas Prantl[1] · Patrick Amann[1] · Christian Krupitzer[2] · Simon Engel[1] · André Bauer[3] · Samuel Kounev[1]

Received: 17 May 2024 / Accepted: 4 September 2024
Published online: 17 September 2024

Abstract
Secure and scalable group communication environments are essential for many IoT applications as they are the cornerstone for different IoT devices to work together securely to realize smart applications such as smart cities or smart health. Such applications are often implemented in Wireless Sensor Networks, posing additional challenges. Sensors usually have low capacity and limited network connectivity bandwidth. Over time, a variety of Secure Group Communication (SGC) schemes have emerged, all with their advantages and disadvantages. This variety makes it difficult for users to determine the best protocol for their specific application purpose. When selecting a Secure Group Communication scheme, it is crucial to know the model's performance under varying network conditions. Research focused so far only on performance in terms of server and client runtimes. To the best of our knowledge, we are the first to perform a network-based performance analysis of SGC schemes. Specifically, we analyze the network impact on the two centralized SGC schemes SKDC and LKH and one decentralized/contributory SGC scheme G-DH. To this end, we used the ComBench tool to simulate different network situations and then measured the times required for the following group operations: group creation, adding and removing members. The evaluation of our simulation results indicates that packet loss and delay influence the respective SGC schemes differently and that the execution time of the group operations depends more on the network situations than on the group sizes.

Keywords Secure Group Communication Schemes · Network conditions · Performance analysis

Abbreviations

GDH	Group Diffie-Hellman
SKDC	Simple key distribution center
LKH	Logical key hierarchy
SGC	Secure Group Communication
IoT	Internet of things
WSN	Wireless sensor network
QoS	Quality of service
MQTT	Message queuing telemetry transport

✉ Thomas Prantl, thomas.prantl@uni-wuerzburg.de; Patrick Amann, patrick.amann@uni-wuerzburg.de; Christian Krupitzer, christian.krupitzer@uni-hohenheim.de; Simon Engel, simon.engel@uni-wuerzburg.de; André Bauer, andrebauer@uchicago.edu; Samuel Kounev, samuel.kounev@uni-wuerzburg.de | [1]Julius-Maximilians-Universität Würzburg, Würzburg, Germany. [2]Universität Hohenheim, Stuttgart, Germany. [3]Illinois Institute of Technology, Chicago, USA.

1 Introduction

The Internet of Things (IoT) is leading to more and more connected devices, and, combined with Big Data, it is a powerful tool to collect and analyze data [1]. A recent report by Statista [2] predicts that the number of connected devices could rise to 29.42 billion by 2030. Compared to 2021, this would represent an increase of more than 160%. This rising number of IoT devices communicating with each other opens up a smart future with a variety of new applications, for instance, in the area of smart health or smart city [3]. As IoT devices often collect and exchange sensitive information with each other, it is essential for emerging smart applications that this data is handled securely. Secure Group Communication (SGC) schemes are an effective way to ensure the security of shared data, as they enable encryption of the n-to-n communication that is typically used for IoT applications [4]. Given their advantages, a large number of SGC schemes have been presented in the literature [4], and a number of works (e.g., [3, 5–7]) have already analyzed their performance on IoT devices. However, these works have in common the fact that the performance analysis of the individual SGC schemes does not take network aspects into account. For example, although the calculation times and storage space requirements of the actors involved or the number and size of the messages sent are considered, they are not analyzed, such as how different network conditions affect the overall time required by an SGC scheme. Considering that IoT devices often form so-called Wireless Sensor Networks (WSNs), large wireless networks of sensors distributed over long distances, it is realistic to assume that not only the hardware resources of IoT devices are limited, but also their network capabilities [8]. E.g., in [9] the authors have shown that in a very simple IoT scenario, in which two IoT devices each read out a sensor and publish this value according to the "fire and forget" principle, packet losses between 5% and 10% can occur. Therefore, it is of utmost importance that the performance analysis of SGC schemes also covers the influence of different network conditions. We intend to close this research gap with this work. Our contributions are concrete:

- Design of a scalable, IoT-typical measurement setup for the evaluation of SGC schemes in different network situations.
- Analysis of the network impact on five SGC schemes of different classes.

The remainder of the paper is structured as follows: Sect. 2 introduces basic concepts that are essential for further understanding. Section 3 provides an overview of related work and highlights the novelty of our contribution. Section 4 presents our typical IoT measurement setup. Section 5 evaluates the five implementations of SGC schemes. In Sect. 6 we discuss potential threats of validity and conclude our paper in Sect. 7.

2 Background

In this section, we first define the term Secure Group Communication (SGC) scheme and present the different classes of SGC schemes. Then, we introduce the MQTT Protocol, a popular and commonly applied protocol for communication in the IoT [3, 10], including its Quality of Services Levels.

2.1 Secure Group Communication schemes

The goal of Secure Group Communication (SGC) schemes is to allow a group to securely agree on a secret group key for encrypting communications with each other. We define Secure Group Communication (SGC) schemes in accordance with [11, 12]. SGC schemes consist of two components: group membership management (GMM) and group key management (GKM). The GMM component provides the operations required to manage group memberships. Specifically, this includes the operations for creating a group, adding a member to the group and removing a member from the group. Probably the simplest way to realize a GMM component would be to create a list of group members, whereby it is possible to delete or add members from the list. All other calculations require that the group members calculate the initial group key or that they update the group key when the group composition changes are managed by the GKM component.

SGC schemes can be divided into the following three classes [4, 13]: Centralized, distributed/contributory and decentralized/hybrid. These three classes are also illustrated in Fig. 1, which emphasizes the conceptual differences between the three classes. In the centralized and decentralized/hybrid SGC schemes, there is a trusted third party, the group controller, controlling the group key generation process. In the centralized SGC schemes, there is no communication between the

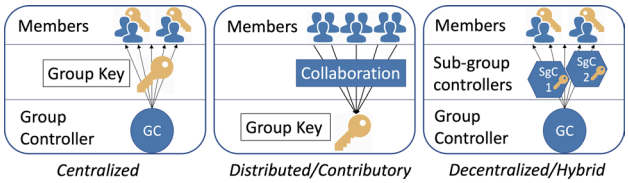


Fig. 1 Illustration of the three classes of SGC schemes [4]

group members regarding the agreement on a group key. In decentralized/hybrid SGC schemes, in contrast, some group members act as sub-group controllers who communicate with other group members on behalf of the group controller. In distributed/contributory SGC schemes, there is no trusted third party controlling the group key generation, but the group members must communicate among themselves to agree on a group key.

2.2 Message queuing telemetry transport protocol

The Message Queuing Telemetry Transport Protocol, or MQTT Protocol for short, is a lightweight machine-to-machine protocol based on a publish-subscribe architecture. Specifically, the architecture consists of a large number of publishers and subscribers, as well as a broker. The publishers represent machines that want to send messages to their subscribers, which in turn are other machines. They do not do this directly but with the help of the broker. To do this, the publishers send their messages to the broker under a specific topic name. The subscribers belonging to the topic can register with the broker for the corresponding topic and thus receive the messages that are published under this topic from the broker. Subscribers can use so-called Quality of Service (QoS) levels to determine how important it is to them that the broker forwards the messages to them. Specifically, the following three QoS levels can be selected: QoS Level 0, the broker only sends the respective messages once and does not check whether they are actually received. QoS Level 1, the broker ensures that the messages arrive at least once, although it can happen that a message arrives several times. QoS Level 2 corresponds to QoS Level 1, except that the broker ensures that messages cannot arrive more than once [3, 14].

3 Related work

Regarding the analysis of SGC schemes, there is literature (e.g., [4, 11, 12, 15]) that surveys and classifies SGC schemes and analyzes their performance in terms of computation time, memory requirements, and communication overhead. However, these costs are only determined theoretically and specified in Landau notation. Concrete network influences and their impact on the performance of the schemes are not considered.

Noteworthy, the articles proposing the respective SGC schemes determined the performance, often using only the Landau Notation. Moreover, if measurements were carried out, no different network situations were taken into account (e.g., [16–25]).

In addition to the surveys and the papers presenting new schemes, there are also a number of articles (e.g., [5, 13, 26–30]) that systematically compare SGC schemes based on measurements. Still, most of them do not consider the network aspect at all, and if they do, they only consider one network setting, namely optimal network conditions.

There are also existing publications (e.g., [6, 7]) that analyze the performance of individual SGC schemes. However, again, no network influences are taken into account.

To the best of our knowledge, none of these studies have included a network-based performance analysis. Our study expands upon this contribution by further assessing the influence of varying network scenarios on the effectiveness of Secure Group Communication schemes.

4 Measurement set up design

In order to analyze the network influence on different SGC schemes in an IoT scenario, we first need an appropriate measurement setup that allows us to conveniently configure (i) how large the groups are and (ii) how the network should behave. Such a measurement setup could be completely realized with real hardware, but this would mean that typical IoT hardware would be required for each group member. In this case, the amount of hardware required and the configuration effort would increase linearly with the size of the group members. As this approach obviously scales poorly, we

decided to use the MQTT benchmark tool ComBench [31] instead. The ComBench tool was developed to conveniently simulate IoT communication in different network scenarios, e.g., via the MQTT protocol, and to measure performance. This corresponds exactly to our use case, as we want IoT devices to communicate in different network situations via MQTT in order to calculate a group key. The functionality of ComBench is illustrated in Fig. 2. ComBench allows the benchmark operator to use a configuration file to specify how many clients should be spawned and how the clients should behave over time. The client's behaviour specifies when they send which message and how they react to received messages. The clients communicate via an MQTT broker, and the benchmark operator can also control how the network should behave over time with a configuration file. In addition, the ComBench tool logs all relevant events and sends the corresponding measured values to the benchmark operator after the benchmark has been carried out.

This fine-grained control makes ComBench a suitable tool for analyzing the network impact on SGC schemes in an IoT scenario. We only need to define the following parameters in the configuration file: (1) The number of clients, (2) when the respective clients send messages, (3) how clients should react to received messages, (4) how large the individual messages are, (5) adapt the performance of the clients to the hardware of the actors involved in the SGC scheme and (6) define the behaviour of the network.

The number of clients is equal to the number of group members. We need n clients for the distributed/contributory SGC schemes. For the centralized or decentralized/hybrid SGC schemes, we need $n + 1$clients since one client must also represent the group controller. In the case of centralized SGC schemes, we later evaluate the following values for n: 50, 100 and 200. For decentralized/contributory SGC schemes, we later evaluate the following values for n: 5, 10 and 20. The configuration when the respective clients send messages differs for centralized and decentralized/hybrid SGC schemes. When a group member is added or removed, the respective member sends a corresponding request to the group controller. In the case of centralized SGC schemes, the group controller then sends a message to all group members. In contrast, in decentralized/hybrid SGC schemes, the group controller first sends a message to the subgroup controllers, which in turn sends messages to the group members. Group creation behaves in the same way, except that the step in which a group member sends the first message is skipped. With distributed/contributory SGC schemes, in contrast, this information must be derived from the functionality of the individual schemes. When defining the behaviour of a client in response to a received message, there are two variants: (i) the client calculates parameters for itself from the message, or (ii) it processes the message and then sends a new message. In both cases, there is a calculation phase, which we define as an IDLE pause in ComBench. This pause is exactly as long as the calculations would have taken. For the parameter settings (4) and (5), we have to define the size of the respective messages and simulate the performance of typical IoT hardware. For our setup, the latter means specifying the length of the IDLE pauses.

In order to be able to specify the parameters for (4) and (5), we decided to measure different group sizes in a real measurement setup and determine the required parameters based on these measurements. To do this, we first had to select the hardware for our IoT scenario. Specifically, this meant that we had to select hardware for both the group members,

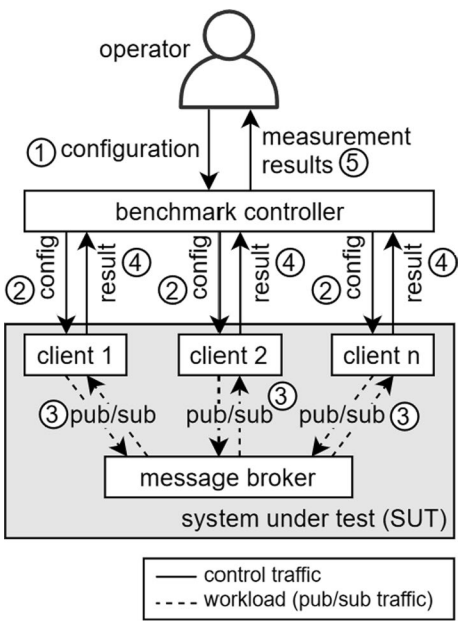


Fig. 2 Architecture of the MQTT benchmark tool ComBench [31]

which represent IoT devices, and, if required by the SGC scheme, for the group controller. For the group members, we selected the ESP32 microcontroller, as it is often used in the literature (e.g., [3, 5–7]) for IoT measurement set-ups. For the group controller, we use a desktop PC with an Intel Core i5 6500 with 3.20 GHz as CPU, 8 GBytes DDR4-2133 RAM and an NVIDIA GeForce GTX 1070. We chose this setup as it is often assumed in the literature that the hardware for the group controller is significantly more powerful than that of the group members [5, 6]. This means that we only still have to define the parameter (6), the behaviour of the network. To do this, we need to specify how the packet loss and the delay should behave over time. For our evaluation, we assume that both the packet loss and the delay are stable over time during a measurement. With regard to packet loss, we consider the following cases: 0%, 5% and 10%. With regard to delay, we consider the following situations: 0ms and 100ms.

The group members and, if available, the group controller communicate with each other via MQTT. We can choose the hardware of the MQTT broker as we wish, as we can set its performance in ComBench using the Packet Loss and Delay parameters. To determine the parameters required for ComBench, we only needed one ESP32 for groups of any size in our real-world measurement set-up. This is because we are only interested in the calculation times of a single group member. So, in the case of centralized SGC schemes for different group sizes, we let the group controller calculate all the parameters needed for each group member. The group controller sent its necessary parameters to our ESP32, which then calculated the group key. Since the calculations of the group members in the case of centralized SGC schemes run independently of each other and only with different numbers, it is sufficient to measure the calculation times of a single group member. A similar procedure can be used for decentralized/hybrid SGC schemes. In this case, the ESP32 would act both as a subgroup controller and as a group member. On the ESP32, first, the calculations of the subgroup controller and then those of a group member would be executed one after the other, and the time required would be measured in each case. For distributed/contributory schemes, we also only need one group member. This is because we can run and simulate the remaining group members on the broker's hardware as corresponding scripts, as we only need to measure the calculation times on real hardware for one group member.

We realized the software of the group controller and the group members with Python 3 or Micropython. The concrete implementation details of the schemes can be found in our paper [26]. The keys required by the SGC schemes were always 128 bits long, which is recommended by the National Institute of Standards and Technology (NIST) [32]. These keys are used to decrypt the messages transmitted in the LKH and SKDC schemes. In the case of the G-DH scheme, the scheme does not require the key agreement messages to be encrypted in order to transport confidential information. In addition, we ensure the authenticity of messages by means of so-called massage authentication codes. Regarding the security model, we have thus ensured the authenticity and confidentiality of the messages for each scheme.

5 Evaluation

In this section, we first present the selection of SGC schemes analyzed in more detail and the specific network conditions. Then, we present the measurements of the centralized SGC schemes and the results of the decentralized SGC scheme. Afterward, we compare the considered SGC schemes with each other and explain the corresponding results.

5.1 SGC scheme and network parameter selection

To select the SGC schemes we wanted to evaluate, we had to decide between investigating many SGC schemes superficially or analyzing a few in detail. We opted for the latter, allowing us to better emphasize the influence of the network on the required times and highlight the importance of network aspects in the performance analysis. We have also decided to implement SGC schemes for the two extreme variants, centralized and distributed/contributory since we want to take the different classes of SGC schemes into account. Specifically, we have implemented the two centralized SGC schemes SKDC [16] and LKH [21]. The SKDC scheme is probably the simplest way to implement a centralized SGC scheme. With SKDC, the group controller generates the group key and encrypts it individually for each group member. A more complex approach to realizing a centralized SGC scheme is LKH, which arranges the group members in a tree structure in order to distribute the group key more efficiently. For the LKH scheme, we have considered all three possible implementation variants (Group-Oriented, User-Oriented and Key-Oriented). These three different implementation versions of LKH do not change its functionality, which allows us to directly compare them with the SKDC scheme. In addition, we also implemented the distributed/contributory SGC scheme GDH [17], which extends the Diffie-Hellman key agreement protocol for 2 parties to groups. For the centralized SGC schemes, we consider exemplary sizes of 50, 100, and 200 members and

groups of 5, 10, and 20 for the distributed/contributory SGC schemes. When selecting the network parameters, we made the same decision as for the SGC schemes and preferred to evaluate a few settings thoroughly, which is why we selected the values 0%, 5%, and 10% as packet loss and 0 ms and 100 ms as delay.

We also need to define the used QoS level. As we are looking at network conditions in which packets can be lost, we have to select the QoS level so that lost packets can be resent; otherwise, the group members would wait indefinitely for lost messages. This leaves us with only QoS Levels 1 and 2 to choose from. For space reasons, we have again decided to analyze one level very precisely rather than present superficial measurements for two levels. We have, therefore, opted for QoS Level 2, which does not flood the network with duplicated messages.

Our restriction on the selection of schemes and network parameters represents a clear limitation of our work, but we would like to point out that for this parameter selection, we had already carried out 2700 measurements to measure each parameter setting 30 times.

5.2 Centralized group encryption schemes

For the evaluation of the centralized SGC schemes, we first evaluate the group creation process for all schemes and then the addition and removal of members.

5.2.1 Group creation

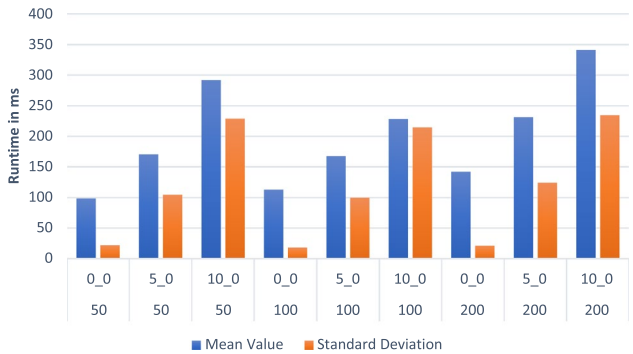
Regarding group creation for centralized SGC schemes, we first consider the SKDC scheme and then the LKH scheme in its three variants.

The two Fig. 3a, b represent the group creation times of the SKDC scheme with the corresponding network parameters. The labels 50, 100, and 200 on the x-axis refer to the group sizes that were tested. The second label *x_y* on the x-axis refers to the packet loss and the delay. Here, *x* stands for the packet loss and *y* for the respective delay.

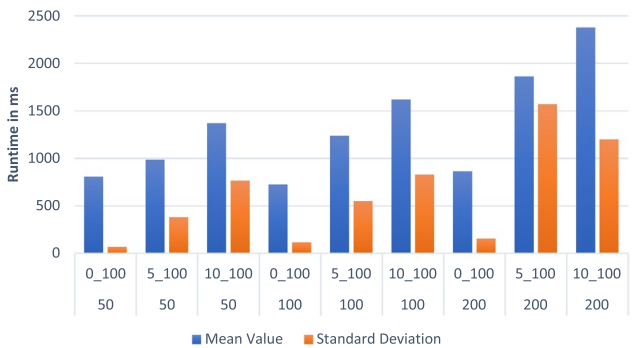
It is evident in both diagrams that the runtime increases with increasing packet loss. If we look at Fig. 3a, for the measurement 0_0 with a group size of 200, the runtime is 142.172 ms, while with a packet loss of 10%, it increases to 340.881 ms. A plausible explanation for this is that with a high packet loss, packets are lost and must be retransmitted, which increases the runtime. At the same time, the standard deviation also increases when the packet loss rate increases.

A comparison between 50 and 100 or 100 and 200 group members indicates a slight increase in runtime when considering all network parameters. One might assume that the runtime increases more with increasing group size. Nevertheless, discrepancies become apparent when comparing group sizes of 50 and 100. When a packet loss rate is introduced, the runtimes with a group size of 100 are even lower than those with a group size of 50.

If we examine Fig. 3b, we can derive identical conclusions. Here, it is particularly noticeable that the runtime increases significantly even with the introduction of a small delay of 100ms. Compared to a runtime of 340.881 ms without delay, this is 2377.318 ms with delay for a group size of 200. Even with a group size of 50 without packet loss and with a delay,



(a) SKDC Group Creation (Delay 0ms).



(b) SKDC Group Creation (Delay 100ms).

Fig. 3 SKDC: group creation. **a** Mean and standard deviation of the time required by the SKDC scheme to create a group without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean and standard deviation of the time required by the SKDC scheme to create a group with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)

the runtime is higher than with a packet loss of 10 % without delay. This study indicates that introducing a delay has a more detrimental impact on runtime compared to a packet loss rate. Both factors, in combination, show clear differences.

Next, we evaluate the group creation process time of the LKH scheme, illustrated in Fig. 4a. All three strategies of LKH are combined in one graph in order to illustrate the differences directly.

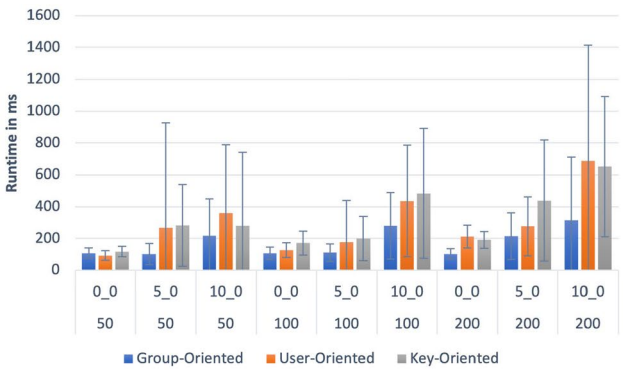
We begin by examining a few general outcomes that are evident from the graphs. In all three methods, with some exceptions, an increase in packet loss rate results in a higher runtime, as anticipated. Correspondingly, the standard deviations also increase since the packet loss rate leads to an increased dispersion of the runtimes. Also, with these algorithms, it can be seen that even a small delay of 100ms leads to a higher runtime than a high packet loss of 10%. In this regard, it can be stated that for the group sizes considered, the delay has a stronger negative effect than a packet loss. Nevertheless, for larger group sizes, this scenario may be altered, as a packet loss rate generates diverse fluctuations, whereas delay predominantly triggers a consistent and predictable fluctuation.

When comparing the algorithms under optimal network conditions, we see that Group-Oriented LKH has the lowest runtime on average, while the Key-Oriented version of LKH has the highest. This is a logically understandable outcome, as with Group-Oriented LKH, only one broadcast message is sent to the clients for group creation. However, with the other two strategies, a message is required for each client as each key must be encrypted with the client's individual key. With the Key-Oriented version of LKH, there is additional overhead because each key must be encrypted individually. On the one hand, this results in a higher runtime for the server. At the same time, the payload of the message increases, so the transmission time of the message also increases. It is important to mention that the runtimes at the Key Server are still very low for the group sizes considered. That is, the majority of the total runtime arises from the transmission of the messages. Accordingly, the fluctuations that can be observed in Fig. 4a are standard.

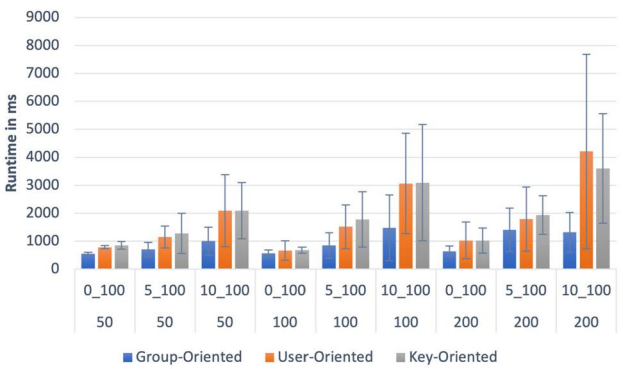
By including a packet loss rate, a few outliers become visible. For example, the runtime for User-Oriented LKH is slightly higher than the Key-Oriented version of LKH with a group size of 50 and a packet loss of 10%. Under optimal network conditions, however, a tendency emerges that was also expected. This tendency is also more strongly supported in Fig. 4b. Here, it can be seen that Group-Oriented LKH performs best on average, while the Key-Oriented version of LKH has the highest runtime. Especially under poor network conditions, Group-Oriented LKH has the significant advantage that only one message needs to be sent. This assumption is supported by our observation that server overhead is lowest for Group-Oriented LKH messaging and highest for Key-Oriented LKH messaging. The number of messages is also a factor, as previously highlighted.

5.2.2 Member addition

After evaluating the group creation process, we analogously evaluate the time it takes to add members for the two centralized SGC schemes, SKDC and LKH.

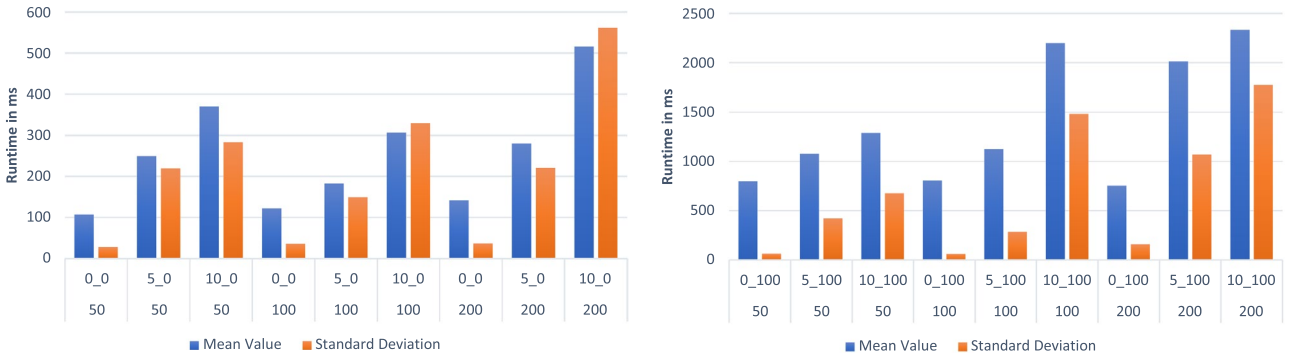


(a) LKH Group Creation (Delay 0ms).



(b) LKH Group Creation (Delay 100ms).

Fig. 4 LKH: group creation. **a** Mean and standard deviation of the time required by the LKH scheme in the group-oriented, user-oriented or key-oriented variant to create a group without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean and standard deviation of the time required by the LKH scheme in the group-oriented, user-oriented or key-oriented variant to create a group with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)



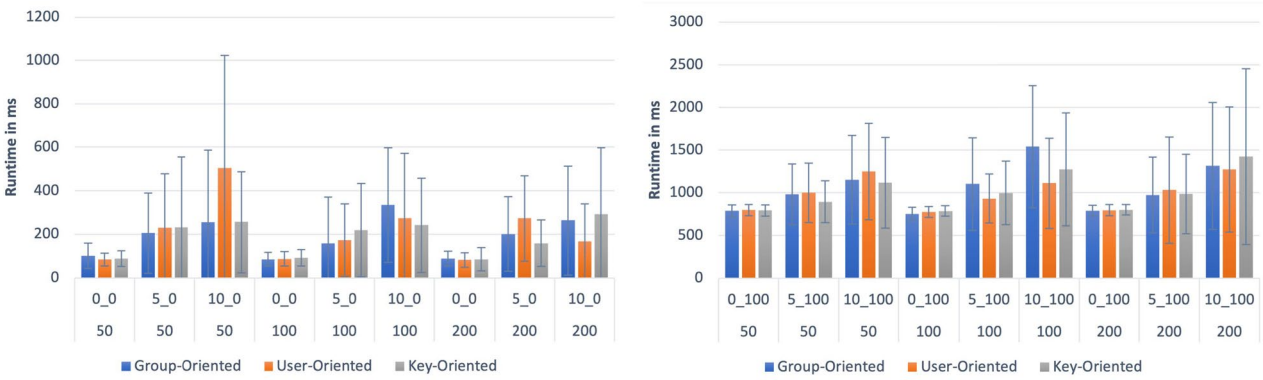
(a) SKDC Join (Delay 0ms). (b) SKDC Join (Delay 100ms).

Fig. 5 SKDC: join operation. **a** Mean and standard deviation of the time required by the SKDC scheme to add a user to the group without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean and standard deviation of the time required by the SKDC scheme to add a uster to the group with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)

The Fig. 5a, b show the time needed by the SKDC scheme to add members to a group. The results are very similar to the group creation. The runtimes increase with increasing packet loss rate. Likewise, a delay has a stronger effect on the runtime than a packet loss rate. For example, with a group size of 200 and a packet loss rate of 10%, the runtime without delay is 516.223 ms, but with a delay of 100 ms without packet loss, it is 795.691 ms, even though the group size is only 50. Apart from a few outliers, the measured values show that the runtime increases with increasing group size. The outliers can be explained by the fact that the network is subject to fluctuations, especially when packet loss rates are inserted.

Next, we evaluate the times needed by the LKH scheme to add members to a group. Two graphs were created to compare the strategies, one for the measurements without delay (Fig. 6a) and one for those with a delay (Fig. 6b).

As with all other measurements, the runtime and the standard deviation increase with increasing packet loss, as expected. Noteworthy, the runtime at the server for a join operation and the considered group sizes are very short, less than one second. Thus, the presented runtime primarily includes transmission times, which account for the majority of the total runtime. It is also important to note that all three algorithms generate an equal number of messages for a join operation in our environment. A message is always generated for the joining client, and a broadcast message is sent to the other members. Therefore, the runtimes are expected to be similar, which is confirmed by the two graphics. Under optimal network conditions, the runtime for all three algorithms and the three group sizes is almost identical. Under poor network conditions, there is otherwise no tendency for one algorithm to work better than



(a) LKH Join (Delay 0ms). (b) LKH Join (Delay 100ms).

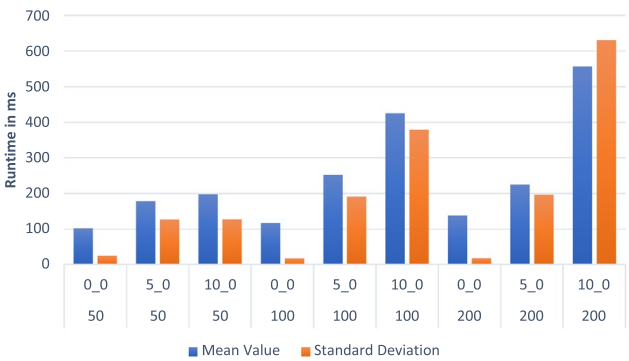
Fig. 6 LKH join operation. **a** Mean and standard deviation of the time required by the LKH scheme in the group-oriented, user-oriented or key-oriented variant to add a uster to the group without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean and standard deviation of the time required by the LKH scheme in the group-oriented, user-oriented or key-oriented variant to add a user to the group with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)

another. The runtimes under optimal conditions are, in principle, almost identical for small group sizes. Under poor network conditions, fluctuations can occur, so sometimes, one algorithm works better than the other. Additionally, the runtime duration experiences a slight shift as the packet loss rate rises, with or without latency. In the event of group creation, runtimes increased by over four times in some cases when both packet loss and delay were included. Figure 6b shows that the values do not even double by inserting a packet loss. This can be attributed to the small number of messages required for a join operation compared to a group creation or leave operation. Especially under poor network conditions, it becomes apparent that the runtime of a join operation is significantly shorter than that of a group creation. From the functioning of the algorithm, it is clear that the transmission times for a join operation for the test environment under consideration are independent of the group size since identical messages are sent regardless of the group size. Since message transmission is associated with fluctuations, the graphics do not clearly show this. However, if we look at optimal conditions, we see almost identical runtimes. Even in Fig. 6b, runtimes for different group sizes are nearly indistinguishable.

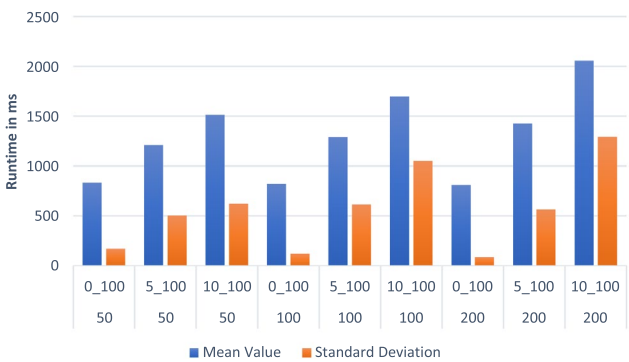
5.2.3 Member revocation

The times needed to revoke a member by the SKDC scheme are illustrated in Fig. 7a, b. The transmission times also increase here with an increase in the packet loss rate. Likewise, the standard deviation increases with an increase in the packet loss rate, as some packets have a longer transmission time than others. This results in a higher dispersion in the measured values. As with leave and group creation, the delay again has a more negative effect on the runtime than packet loss. The measured values indicate an increase in the runtime as the group size grows.

The time needed by the LKH scheme to revoke a member is illustrated in Fig. 8a, b. Firstly, it can be stated that the runtimes and standard deviations rise with increasing packet loss rates for all three strategies. In the case of a leave operation, the runtimes at the Key Server now play a greater role. As described in detail above, these range from just under one millisecond to just over one second, depending on the group size and algorithm. If we examine the two charts, it is evident that the total runtime is mainly influenced by the transmission time of the messages. In the case of ideal network conditions, Group-Oriented LKH delivers the best average runtimes, while Key-Oriented LKH has the highest runtime. Especially under poor network conditions, which are shown in Fig. 8b, this claim is substantiated. Again, this is because Group-Oriented LKH leave generates only one re-key message, while the other two strategies require a message for each client. This results in a longer runtime, particularly under poor network conditions. Therefore, with a high packet loss rate, significant fluctuations in delays can occur, especially with larger group sizes. This is evident even with small group sizes but still within the bounds of discretion. The graphs also show higher runtimes for a leave operation than for a join. This is particularly visible under poor network conditions, as the large number of messages in a leave operation compared to a join operation becomes particularly apparent here.

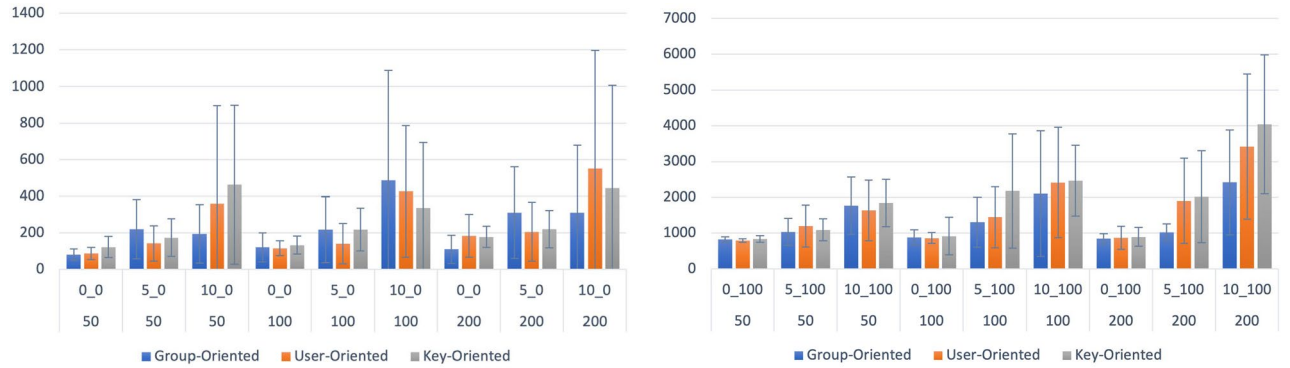


(a) SKDC Leave Operation (Delay 0ms).



(b) SKDC Leave Operation (Delay 100ms).

Fig. 7 SKDC leave operation. **a** Mean and standard deviation of the time required by the SKDC scheme to remove a member from the group without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean and standard deviation of the time required by the SKDCscheme to remove a member from the group with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)



(a) LKH (Delay 0ms). (b) LKH (Delay 100ms).

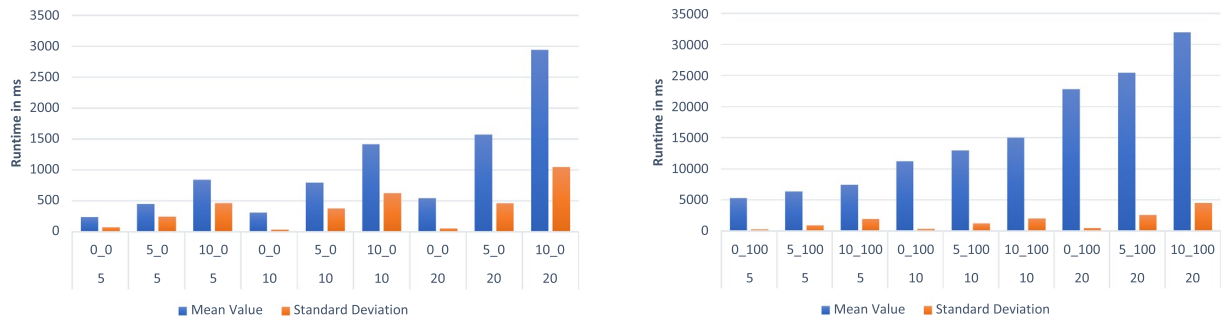
Fig. 8 LKH leave operation. **a** Mean and standard deviation of the time required by the LKH scheme in the group-oriented, user-oriented or key-oriented variant to remove a member from the group without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean and standard deviation of the time required by the LKH scheme in the group-oriented, user-oriented or key-oriented variant to remove a member from the group with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)

5.3 Decentralized group encryption schemes

We now present our measurements of the decentralized algorithm G-DH. We proceed analogously to the centralized SGC schemes and first evaluate the group creation process, followed by the addition and removal of members.

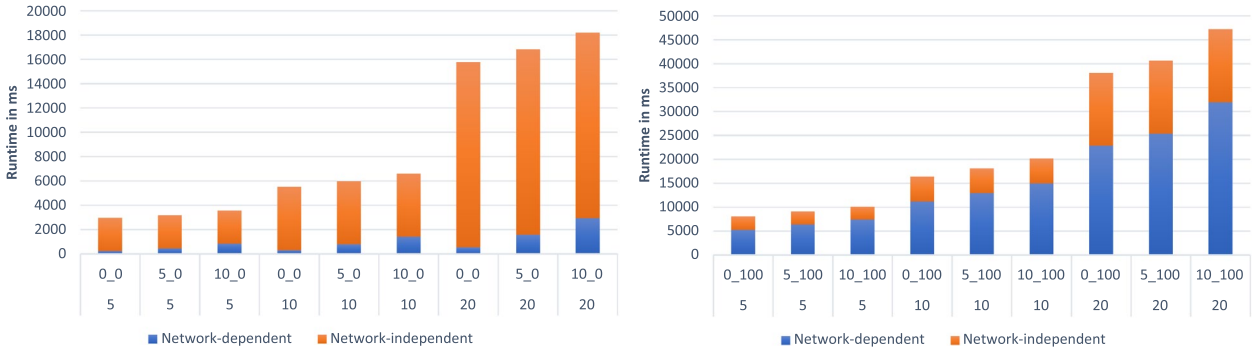
5.3.1 Group creation

Figure 9a displays the runtimes and standard deviation for all measured values at a network delay of 0ms. Figure 9b illustrates the values with a network delay of 100ms. With increasing group size, the total group creation time increases, regardless of the selected network parameters. This is because the number of messages increases linearly with the number of clients. Regardless of the delay, the runtime increases with increasing packet loss, as packets are lost and have to be retransmitted. At the same time, this also leads to a higher dispersion around the mean value, as some packets are lost and, therefore, lead to a longer runtime, while other packets can be transmitted without losses. Comparing the two graphs, it becomes clear that packet loss has a smaller effect on the runtime than a network delay. If we look at the runtime with a delay of 0 ms and compare the values for the packet loss of 0 % and 10%, we find



(a) Group Creation Message Transmission Runtime (Delay 0ms). (b) Group Creation Message Transmission Runtime (Delay 100ms).

Fig. 9 G-DH: group creation message transmission runtime. **a** Mean and standard deviation of the time required by the G-DH scheme to transmit all messages during the group creation process without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean and standard deviation of the time required by the G-DH scheme to transmit all messages during the group creation process with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)



(a) Group Creation Total Runtime (Delay 0ms). (b) Group Creation Total Runtime (Delay 100ms).

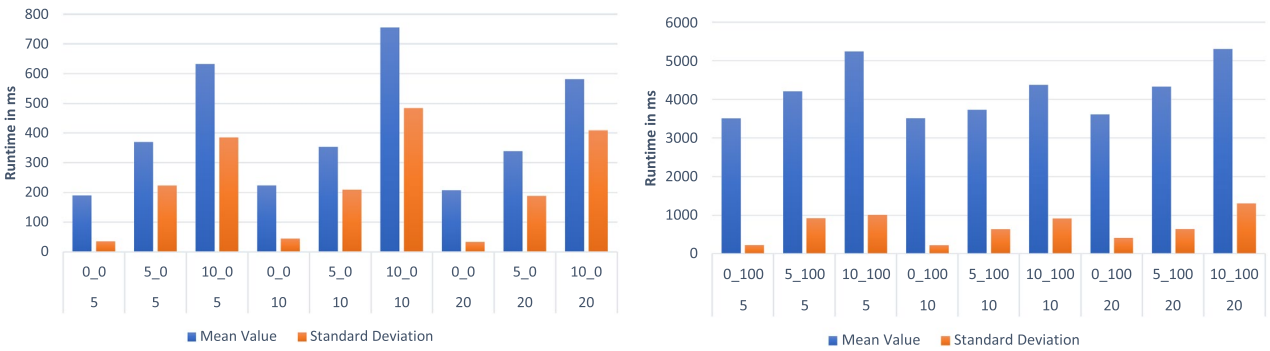
Fig. 10 G-DH: group creation total runtime. **a** Mean of the newtork-dependent and network-independent time required by the G-DH scheme to create a group without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean of the newtork-dependent and network-independent time required by the G-DH scheme to create a group with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)

a runtime of 537.113 ms compared to 2942.719 ms for a group size of 20. As soon as a delay is added, the runtimes increase enormously to 22817.9 ms and 31975.445 ms, respectively.

Figure 10a (delay 0ms) and Fig. 10b (delay 100ms) illustrate the comparison of network-independent and network-dependent measurements for the examined network parameters. With a delay of 0 ms, it is evident that the clients' processing time is responsible for the majority of the total runtime. Consequently, sending messages plays a relatively negligible role. However, if a delay is added, the exchange of messages clearly dominates with regard to the total runtime. From a total runtime of 18192.627 ms with a packet loss of 10 % without delay, this has increased to 47225.364 ms after inserting a delay (group size 20). The share of the network-dependent delay increased from 16.18 % to 67.71 %. A high delay is critical for this algorithm with regard to group creation. Overall, the runtime is not linear since the computational effort for the clients increases non-linearly with increasing group size.

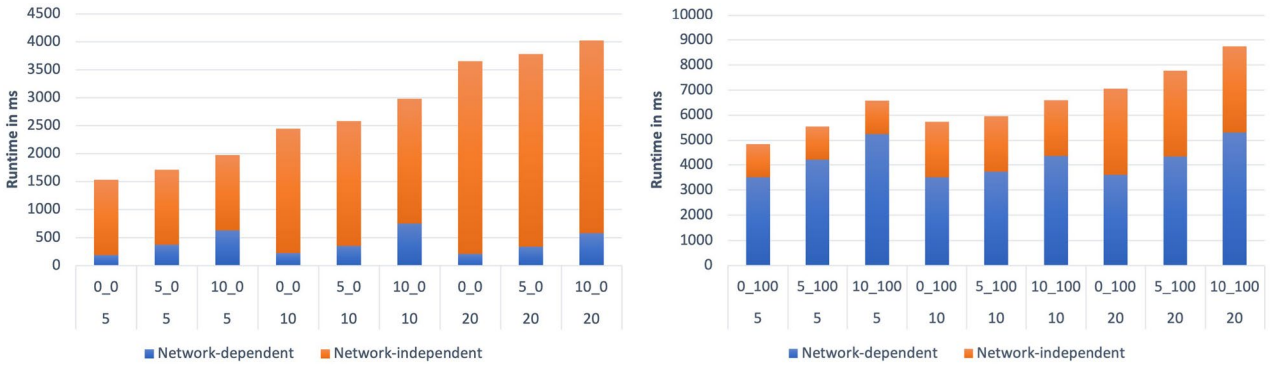
5.3.2 Member addition

As with the group creation, Fig. 11a, b present the run times for a delay of 0ms and 100ms separately. The figures also demonstrate that the runtime increases with increasing delay or packet loss. This can be explained for the same reasons as for the group creation. Furthermore, the graph indicates that the run time is potentially unaffected by group size. With slight variations, the runtimes for the different group sizes are identical for a fixed delay. With a packet loss of 10 %, the



(a) Join Message Transmission Runtime (Delay 0ms). (b) Join Message Transmission Runtime (Delay 100ms).

Fig. 11 G-DH: join message transmission runtime. **a** Mean and standard deviation of the time required by the G-DH scheme to transmit all messages during the join process of a new group member without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean and standard deviation of the time required by the G-DH scheme to transmit all messages during the join process of a new group member with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)



(a) Join Total Runtime (Delay 0ms). (b) Join Total Runtime (Delay 100ms).

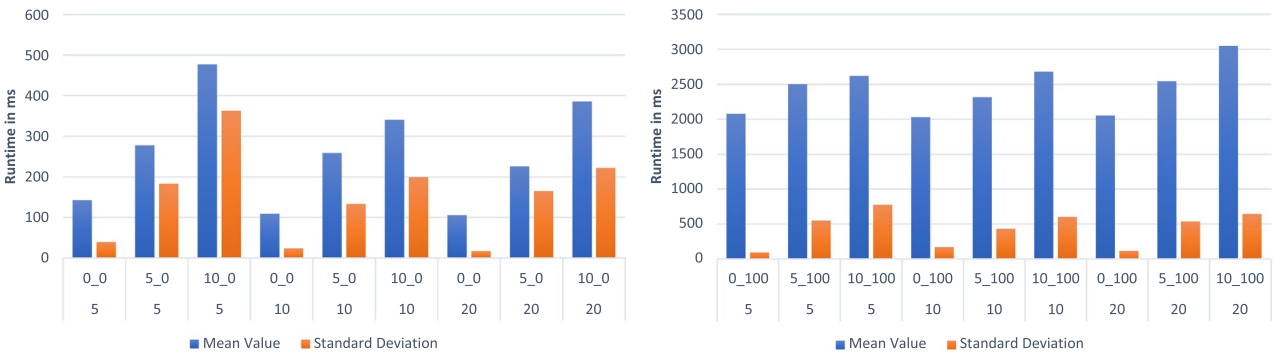
Fig. 12 G-DH: Join operation total runtime. **a** Mean of the newtork-dependent and network-independent time required by the G-DH scheme to add a member to the group without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean of the newtork-dependent and network-independent time required by the G-DH scheme to add a member to the group with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)

deviation between the different group sizes is stronger. With a high packet loss rate, the loss of the packet is dependent on chance, which is why fluctuations are normal here. However, as the group size increases, the payload of the message increases. For the tested group sizes, this has no influence on the runtime. For larger group sizes, however, this should be taken into account.

If we examine the entire runtime of a join operation in Fig. 12a, b, the conclusions are similar to those of group creation. The insertion of a delay has a much stronger negative influence on the runtime than a packet loss rate. It should also be mentioned that the runtime of a join operation is generally lower than that of a group creation since significantly fewer messages have to be exchanged. In addition, there are always only two clients that have to perform a forward or backward pass. These are the joining client and the last client within the group, who has a special role. The graphics demonstrate that the join operation's runtime is dependent on the group size. The clients' calculation effort increases as the group size increases.

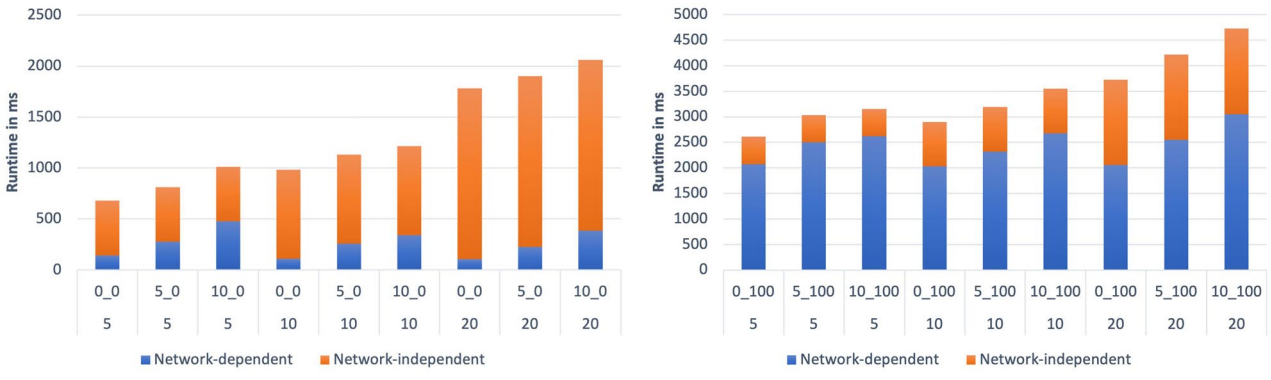
5.3.3 Member revocation

Figure 13a, b again show the results for a delay of 0 ms and 100 ms, respectively. It is evident that packet loss adversely affects the runtime. The runtimes for different group sizes are almost identical. The group size only has an impact on the message's payload. Since fewer messages are sent during a leave operation compared to a join, any influence is reduced



(a) Leave Message Transmission Runtime (Delay 0ms). (b) Leave Message Transmission Runtime (Delay 100ms).

Fig. 13 G-DH: leave message transmission runtime. **a** Mean and standard deviation of the time required by the G-DH scheme to transmit all messages during the revocation process of a group member with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean and standard deviation of the time required by the G-DH scheme to transmit all messages during the revocation process of a group member without delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)



(a) Leave Total Runtime (Delay 0ms). (b) Leave Total Runtime (Delay 100ms).

Fig. 14 G-DH: leave operation total runtime. **a** Mean of the newtork-dependent and network-independent time required by the G-DH scheme to revoke a member from the group without a delay. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*). **b** Mean of the newtork-dependent and network-independent time required by the G-DH scheme to revoke a member from the group with a delay of 100 ms. (Note: The labelling *x_y z* on the *x*-axis stands for a packet loss of *x*%, a delay of *y* ms and a group size of *z*)

even further. The higher runtime with a group size of 5 (Packet Loss: 10 %, Delay: 0 ms) is also noticeable here in comparison to larger group sizes. However, the high fluctuations are caused by the high packet loss and general network variance. Additionally, Fig. 13b demonstrates that a delay of 100 ms significantly affects the runtime negatively, even more so than a high packet loss of 10%.

The total runtimes were also determined here. These are illustrated in Fig. 14a, b. The conclusions are identical to those of the join operation. In addition, it should be mentioned that for the measured group sizes, the runtime of a leave operation is about half that of a join operation.

5.4 Discussion

In this section, we compare all the algorithms and evaluate their pros and cons. We also provide guidance to assist developers in selecting the most suitable algorithm.

5.4.1 Group creation

During group creation, it was observed that SKDC is faster than the following two implementation versions of the LKH scheme: Key-Oriented and User-Oriented. However, for the group sizes under consideration, the Group-Oriented LKH version achieved better runtimes than SKDC. This observation remained consistent under both optimal and poor network conditions. The decentralized algorithm G-DH delivers significantly poor runtimes here. The high runtimes become very apparent, especially under poor network conditions. Even with group sizes ranging from 5 to 20, the runtimes are notably higher than those of centralized approaches with group sizes of 50 to 200. The main cause of this runtime, under optimal conditions, is the calculations performed at each client. The centralized protocols clearly have the advantage of a low runtime for the clients and the server. In addition, G-DH is adversely affected by delay as each client receives a message and forwards it, whereas, with centralized protocols, mainly broadcast messages are sent. A medium delay combined with extensive packet loss causes a runtime of nearly 60 s. Given large group sizes, it is likely that G-DH will probably no longer be usable under poor network conditions.

5.4.2 Member addition

It is clearly visible that the server's effort for SKDC is significantly greater than that of the tree-based approach, regardless of the strategy. This is a key benefit of the tree-based approach. Additionally, only a small number of re-key messages are required for the tree-based approaches, leading to low transmission time. Even under suboptimal network conditions, the total runtime is barely affected in the case of the tree-based approaches. With small group sizes, a delay has a more detrimental effect on the overall runtime for both algorithms. However, this may alter with large group sizes, as

a significant packet loss can cause considerable fluctuations in the runtime, whereas a delay usually results in constant latencies. If we look more closely at the join operation with G-DH, it is again evident that this causes the highest runtime. Here, the runtime is higher even with significantly smaller group sizes than with the centralized methods for larger group sizes. Above all, a high delay can cause significant problems here, leading to enormous runtimes for larger group sizes.

5.4.3 Member revocation

For the Leave operation, it can be observed that SKDC has lower server overheads compared to the tree-based approach across all three strategies. Under ideal network conditions, the runtime of the Group-Oriented version of LKH is still very close to that of SKDC. However, in suboptimal networking scenarios, the difference in runtime widens, affirming SKDC's superiority. In line with expectations, the decentralized algorithm exhibits a longer runtime than the centralized protocols. However, the leave operation causes the shortest runtimes in G-DH compared to the group creation and join operation. Here, a delay has a more detrimental effect on the runtime than a packet loss for all of the algorithms for the group sizes considered.

6 Threats to validity

Like any scientific work, our work also has limitations. These include, for example, the assumptions we made for the configuration of ComBench. For our simulation, we assume that the network conditions do not change during the measurements. In reality, however, network conditions can certainly fluctuate over time. A simple example of this would be a smart city use case where the internet connection of an IoT device is briefly interrupted by passing vehicles. We also assume that all IoT devices use the same hardware for our simulation, which is not necessarily the case in reality. The validity of the evaluation of the simulation results is, therefore, limited to the specific network conditions and hardware that we used for our measurements. As we have only considered a limited number of group sizes, our statements are, of course, also only valid for the specific sizes considered.

7 Conclusion

With the increasing number of IoT devices, secure communication between them is essential. Secure Group Communication (SGC) schemes play a pivotal role in enabling encrypted communication, as is common in IoT scenarios. Studies analyzing such schemes and even the articles proposing them often overlook the impact of network conditions, a critical consideration given the widespread use of IoT devices in WSNs. This research aims to address this gap by comprehensively evaluating the performance of SGC schemes, taking into account the constraints imposed by varying network conditions in IoT environments.

More specifically, we employed ComBench to create a scalable measurement setup mirroring typical IoT scenarios, facilitating the evaluation of SGC schemes across diverse network conditions. Our analysis encompassed five SGC schemes from both centralized and decentralized categories, delving into server runtimes, client runtimes, and transmission times. By introducing variations in delay and packet loss to simulate adverse network conditions, our findings highlight the significant impact of poor network conditions on runtime, particularly noticeable in smaller group sizes where message transmission time dominates. Under poor conditions, the decentralized approach experiences substantial runtime increases, making a centralized protocol more preferable. Additionally, observations indicate that SKDC is less suited for dynamic groups due to its extensive messaging, making a tree-based approach, especially in join operations, more efficient and suitable for such scenarios.

As part of our future work, we aim to expand this study by incorporating additional SGC schemes. More specifically, we plan to incorporate the results of our benchmark for SGC schemes [26] into our simulation. This will allow us to extend our simulation not only with further centralized and distributed/contributory SGC schemes but also with decentralized/hybrid SGC schemes. Additionally, we want to integrate our work in the field of attribute-based encryption schemes so that we can compare attribute-based encryption schemes with SGC schemes [33–35]. We also plan to extend our simulation with more heterogeneous IoT devices. Recognizing the existing research gap, particularly in network-based analysis, our prospective work may involve integrating further network parameters, such as network bandwidth, into the evaluation. This exploration could encompass investigating the impact of diverse payload sizes on transmission

time. Additionally, there is potential for examining SGC schemes under alternative evaluation criteria, measuring power consumption, and testing robustness against potential attacks.

Author contributions The authors contributed equally to the paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability All presented data has been generated using the ComBench simulation tool.

Code availability The code of ComBench ist available here https://github.com/DescartesResearch/ComBench.

Declarations

Competing interests The authors declare that they have no Competing interests.

References

1. Rose K. et al., The internet of things: an overview understanding the issues and challenges of a more connected world. 2015. https://api.semanticscholar.org/CorpusID:9217381. Accessed 12 Aug 2024.
2. Statista. Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030. 2023. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/. Accessed 12 Aug 2024.
3. Prantl T, Iffländer L, Herrnleben S, Engel S, Kounev S, Krupitzer C. Performance impact analysis of securing MQTT Using TLS, in Proceedings of the ACM/SPEC International Conference on Performance Engineering (ACM, 2021), ICPE '21. https://doi.org/10.1145/3427921.3450253.
4. Prantl T, Zeck T, Bauer A, Ten P, Prantl D, Yahya AEB, Ifflaender L, Dmitrienko A, Krupitzer C, Kounev S. A survey on secure group communication schemes with focus on iot communication. IEEE Access. 2022;10:99944–62. https://doi.org/10.1109/access.2022.3206451.
5. Prantl T, Ten P, Iffländer L, Herrnleben S, Dmitrenko A, Kounev S, Krupitzer C. Towards a group encryption scheme benchmark: a view on centralized schemes with focus on IoT, in Proceedings of the ACM/SPEC International Conference on Performance Engineering (ACM, 2021), ICPE '21. https://doi.org/10.1145/3427921.3450252.
6. Prantl T, Ten P, Ifflander L, Dmitrenko A, Kounev S, Krupitzer C. Evaluating the performance of a state-of-the-art group-oriented encryption scheme for dynamic groups in an iot scenario, in 2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) (IEEE, 2020). https://doi.org/10.1109/mascots50786.2020.9285948.
7. Prantl T, Engel S, Bauer A, Yahya AEB, Herrnleben S, Ifflander L, Dmitrienko A, Kounev S. An experience report on the suitability of a distributed group encryption scheme for an IoT use case, in 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring) (IEEE, 2022). https://doi.org/10.1109/vtc2022-spring54318.2022.9860762.
8. Nack F. An overview on wireless sensor networks. Vol. 6. Institute of Computer Science (ICS), Freie Universität Berlin; 2010.
9. Pramukantoro ES, Anwari H. An event-based middleware for syntactical interoperability in internet of things. Int J Electr Comput Eng (IJECE). 2018;8(5):3784. https://doi.org/10.11591/ijece.v8i5.pp3784-3792.
10. Perrone G, Vecchio M, Pecori R, Giaffreda R. The day after mirai: a survey on mqtt security solutions after the largest cyber-attack carried out through an army of iot devices, in Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (SCITEPRESS - Science and Technology Publications, 2017). https://doi.org/10.5220/0006287302460253.
11. Sakarindr P, Ansari N. Security services in group communications over wireless infrastructure, mobile ad hoc, and wireless sensor networks. IEEE Wirel Commun. 2007;14(5):8–20. https://doi.org/10.1109/mwc.2007.4396938.
12. Cheikhrouhou O. Secure group communication in wireless sensor networks: a survey. J Netw Comput Appl. 2016;61:115–32. https://doi.org/10.1016/j.jnca.2015.10.011.
13. Prantl T, Bauer A, Ifflander L, Krupitzer C, Kounev S. Recommendation of secure group communication schemes using multi-objective optimization. Int J Inf Secur. 2023;22(5):1291–332. https://doi.org/10.1007/s10207-023-00692-0.
14. Lee S, Kim H, kweon Hong D, Ju H. Correlation analysis of mqtt loss and delay according to qos level, in The International Conference on Information Networking 2013 (ICOIN) (IEEE, 2013). https://doi.org/10.1109/icoin.2013.6496715.
15. Rafaeli S, Hutchison D. A survey of key management for secure group communication. ACM Comput Surv. 2003;35(3):309–29. https://doi.org/10.1145/937503.937506.

16. Li SQ, Wu Y. A survey on key management for multicast, in 2010 Second International Conference on Information Technology and Computer Science (IEEE, 2010). https://doi.org/10.1109/itcs.2010.82.
17. Steiner M, Tsudik G, Waidner M. Diffie-hellman key distribution extended to group communication, in Proceedings of the 3rd ACM Conference on Computer and Communications Security. 1996; 31–37.
18. Gaddour O, Koubâa A, Abid M. Segcom: a secure group communication mechanism in cluster-tree wireless sensor networks, in 2009 First International Conference on Communications and Networking IEEE. 2009; 1–7.
19. Waldvogel M, Caronni G, Sun D, Weiler N, Plattner B. The versakey framework: versatile group key management. IEEE J Sel Areas Commun. 1999;17(9):1614.
20. Ballardie T, Crowcroft J. Multicast-specific security threats and counter-measures, in Proceedings of the Symposium on Network and Distributed System Security (IEEE, 1995). 2–16
21. DeCleene B, Dondeti L, Griffin S, Hardjono T, Kiwior D, Kurose J, Towsley D, Vasudevan S, Zhang C. Secure group communications for wireless networks, in 2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No. 01CH37277), vol. 1 (IEEE, 2001), vol. 1. 113–117.
22. Penrig A, Song D, Tygar D. Elk, a new protocol for efficient large-group key distribution, in Proceedings 2001 IEEE Symposium on Security and Privacy. S &P 2001. (IEEE, 2000). 247–262.
23. Son JH, Lee JS, Seo SW. Topological key hierarchy for energy-efficient group key management in wireless sensor networks. Wirel Pers Commun. 2010;52:359.
24. Burmester M, Desmedt Y. A secure and efficient conference key distribution system, in Advances in Cryptology-EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, May 9–12, 1994 Proceedings 13. Springer. 1995; 275–286.
25. Alohali BA, Vassilakis VG, Moscholios ID, Logothetis MD. A secure scheme for group communication of wireless IoT devices, in 2018 11th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP) (IEEE, 2018). 1–6.
26. Prantl T, Bauer A, Engel S, Horn L, Krupitzer C, Ifflander L, Kounev S. Benchmarking of secure group communication schemes with focus on iot. Discov Data. 2024. https://doi.org/10.1007/s44248-024-00010-6.
27. Adekanmbi O, Omitola O, Oyedare T, Olatinwo S. Performance evaluation of common encryption algorithms for throughput and energy consumption of a wireless system. J Adv Eng Technol. 2015;3(1):1.
28. Amir Y, Kim Y, Nita-Rotaru C, Tsudik G. On the performance of group key agreement protocols. ACM Trans Inf Syst Secur. 2004;7(3):457–88. https://doi.org/10.1145/1015040.1015045.
29. Zheng S, Alves-Foss J, Lee SS. Performance of group key agreement protocols over multiple operations., in IASTED PDCS. 2005; 600–606.
30. Qikun Z, Yongjiao L, Yong G, Chuanyang Z, Xiangyang L, Jun Z. Group key agreement protocol based on privacy protection and attribute authentication. IEEE Access. 2019;7:87085–96. https://doi.org/10.1109/access.2019.2926404.
31. Herrnleben S, Leidinger M, Lesch V, Prantl T, Grohmann J, Krupitzer C, Kounev S. ComBench: a benchmarking framework for publish/subscribe communication protocols under network limitations. New York: Springer International Publishing; 2021. p. 72–92.
32. Barker E. Recommendation for key management:: part 1 - general (2020).https://doi.org/10.6028/nist.sp.800-57pt1r5.
33. Prantl T, Zeck T, Horn L, Ifflnder L, Bauer A, Dmitrienko I, Krupitzer C, Kounev S. Towards a cryptography encyclopedia: a survey on attribute-based encryption. J Surveill Secur Saf. 2023;4(4):129–54. https://doi.org/10.20517/jsss.2023.30.
34. Prantl T, Zeck T, Ifflander L, Beierlieb L, Dmitrenko A, Krupitzer C, Kounev S. Towards a cryptography benchmark: A view on attribute based encryption schemes, in 2022 5th Conference on Cloud and Internet of Things (CIoT) (IEEE, 2022). https://doi.org/10.1109/ciot53061.2022.9766494.
35. Prantl T, Lauer M, Horn L, Engel S, Dingel D, Bauer A, Krupitzer C, Kounev S. Security analysis of a decentralized, revocable and verifiable attribute-based encryption scheme, in Proceedings of the 19th International Conference on Availability, Reliability and Security (ACM, 2024), ARES 2024. https://doi.org/10.1145/3664476.3664487.