2022 IEEE 95th Vehicular Technology Conference (VTC2022-Spring) | 978-1-6654-8243-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/VTC2022-Spring54318.2022.9860762

An Experience Report on the Suitability of a Distributed Group Encryption Scheme for an IoT Use Case

Thomas Prantl, Simon Engel, André Bauer, Ala Eddine Ben Yahya, Stefan Herrnleben,
Lukas Iffländer, Alexandra Dmitrienko and Samuel Kounev
University of Würzburg, Germany
{firstname.lastname}@uni-wuerzburg.de

***Abstract*—The critical component in any IoT application is the communication between devices. This must not only function smoothly, but also be secured. An important step in securing IoT communication is its encryption. However, in order for IoT devices to encrypt their communication with each other, they must first agree on appropriate cryptographic keys. In practice, the generation and distribution of such keys is usually managed by a central authority. However, this centralized approach has the disadvantages that (i) a central authority must be trusted, (ii) the central authority represents a single point of failure, and (iii) the central authority may be far away and thus communication with it takes a long time. To overcome these drawbacks, distributed group key agreement approaches have also been proposed. Since these distributed approaches were not originally developed for IoT devices, their performance on such devices is unknown. Therefore, in this work, we investigate the performance of a distributed group encryption scheme on IoT devices. To this end, we have built a measurement environment for distributed group encryption schemes and compare centralized and distributed group encryption schemes for IoT.**

Our measurements show that under perfect network conditions, the distributed approach performs worse than the centralized approaches in terms of time and memory requirements. However, our measurements also show that the distributed approach allows a group of 5 members to agree on a key in less than a minute. Thus, the distributed method can be used for small IoT groups if agreeing on a key is not time-sensitive.

***Index Terms*—Distributed Group Encryption Scheme, Performance Analysis, IoT**

I. Introduction

In 1999, Kevin Ashton coined the term Internet of Things (IoT) during a presentation at Procter & Gamble [1]. Since then, more and more devices and machines have been communicating more effectively thanks to ever smaller, cheaper, and more powerful components and sensors. Nowadays, a vast network of connected devices collect and analyze data and perform tasks on their own. This network enables many new applications in various domains such as health care, sports, or transportation. One use case from transportation that benefits from the ever-increasing efficiency of IoT is platooning. In a so-called platoon, a group of vehicles drives in a coordinated manner at short intervals behind each other. This short headway can be safely maintained because the vehicles communicate with each other and can thus accelerate or brake simultaneously. Consequently, the platoon's vehicles consume less fuel to maintain traffic flow and due to reduced air resistance.

The essential but also the most critical element in many IoT applications is the smooth communication between the devices. In the case of platooning, communication provides the exchange of various information and commands between vehicles to coordinate the platoon. Since a hacker could hack into the platoon and cause property damage or even endanger human lives through accidents, it is therefore crucial that the communication between the vehicles or generally the communication between IoT devices is encrypted. Another challenge is that due to the flexibility and mobility of IoT devices, IoT devices do not need to know each other beforehand. Therefore, IoT devices must spontaneously agree on a key for encryption/decryption. To tackle this problem, various procedures have been presented in the literature. One approach is the use of a central entity that specifies a key. However, this approach has several problems: (i) A IoT device must trust a central entity; (ii) The central entity may be unreachable (for an IoT device); (iii) The distance between IoT devices and the central instance can be too great so that communication and thus group generation takes longer.

To avoid relying on a single central authority and thus circumventing its disadvantages, a distributed approach to agreeing on a key is required. Consequently, different distributed methods have been presented in the literature. However, these (and the centralized) procedures were not designed for IoT; therefore, their performance on IoT devices is almost unknown. Analogously to our previous papers [2] and [3], where we determined the performance of centralized schemes on IoT devices, we now want to investigate the performance of the distributed scheme proposed by Kim et al. [4] on IoT devices.

More specifically, our contributions are:

- A IoT typical measurement environment for the distributed group encryption scheme, proposed by Kim et al, including workloads and metrics.
- A performance analysis of the distribute scheme [4], proposed by Kim et al., in an Iot scenario.

978-1-6654-8243-1/22/$31.00 ©2022 IEEE

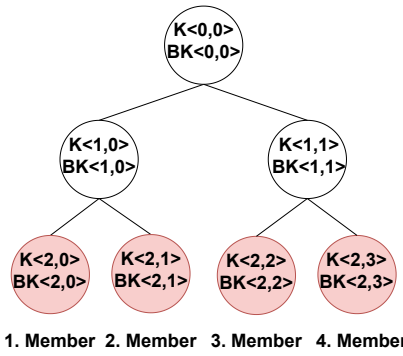

Fig. 1. Illustration of the concept of Kim et al's. [4] scheme, which consists of arranging the group members in a tree that is kept up to date by the group members.

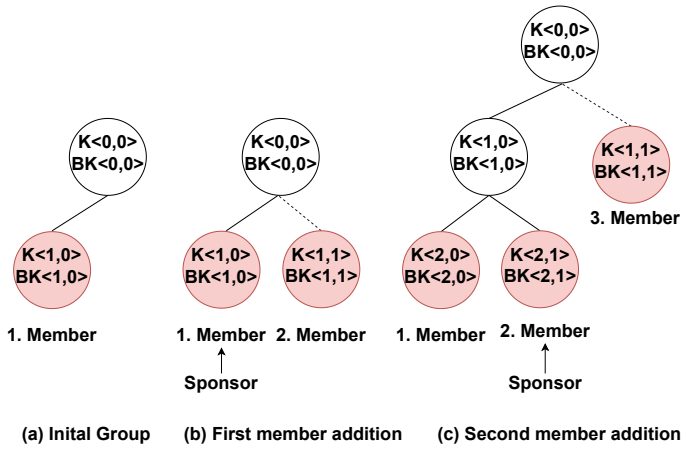

Fig. 2. Illustration of the successive addition of 2 new group members to a group that initially consists of only one member.

- A performance comparison of centralized and distribute group encryption schemes in an IoT scenario.

The remainder of this paper is structured as follows. In Section II, we present the concept of the distributed group encrpytion scheme proposed by Kim et al. We then present our evaluation environment for the distributed scheme in Section III, including a measurment set up, metrics and workloads. In Section IV, we present the performance analysis of the distributed scheme proposed by Kim et al. and compare this scheme with centralized schemes. Finally, we highlight the novelties of our contributions by comparing them with related work in Section V and summarize our paper in Section VI.

II. Background

In this section, we introduce the group management operations of the scheme developed by Kim et al. [4]. Specifically, we present (i) the general concept of the scheme developed by Kim et al., (ii) the initial creation of a group consisting of n members, (iii) the addition and (iv) the revocation of members of this group.

(1) Fundamental Concept: The basic idea of the scheme proposed by Kim et al. [4] is that the members of a group arrange themselves in a tree structure, which allows them to compute a common group key. This tree is kept up-to-date by the group members themselves, i.e. when adding or removing group members, each group member has to update its own individual tree. The update of the tree in case of a new or removed member will be discussed in the following subsections, here we show how the group members can calculate the group key on the basis of a concrete tree. Such an exemplary tree is illustrated in Figure 1, showing that (1) the members of the group form the leaves of the tree and (2) each node has a key $K_{<l,v>}$ and a blinded key $BK_{<l,v>}$. Here, l represents the depth of the respective node in the tree and v represents the position of the node at depth v. A group member chooses its key $K_{l,v}$ itself and can compute its blinded key $BK_{<l,v>}$ by $f(K_{<l,v>})$, where the function f is defined in Equation 1. The exponentiation base α and the prime number p in the function f are predetermined and publicly known. A group member can calculate the key K and hence BK of its parent node using Equation 2. This requires the group member to know the key K of the left child of the parent node and the blinded key BK of the right child of the parent node. Since the ultimate goal of each group member is to calculate the key of the root of the tree $K_{<0,0>}$, each group member must know all the corresponding keys on its way to the root, or have the information to calculate them. (Note: This is always ensured by the way members are added or removed). The root key $K_{<0,0>}$ and a hash function h allow the group members to subsequently compute the group key k_{group} as follows: $k_{group} = h(K_{<0,0>})$. The group key k_{group} can then be used to encrypt or decrypt messages to or from the group.

$$f(x) = \alpha^x \text{mod } p \tag{1}$$

$$K_{<l,v>} = (BK_{<l+1,2v+1>})^{K<l+1,2v>} \text{mod } p \tag{2}$$

(2) Initial Group Creation: Creating a group with n members begins with the first member acting as the sponsor and creating the initial key tree, as shown in Figure 2a. The remaining members are then added individually to the tree and thus to the group. The process of adding a group member is described below.

(2) Member Addition: Adding a new group member starts with the new group member generating its key K and computing its blinded key BK from it. Then the new group member broadcasts a join request to the existing group containing its BK. Then the existing members of the group determine the insertion point for the new member in the key tree, distinguishing two cases: (1) the tree is balanced and (2) the tree is not balanced. In the case of a balanced tree, the new member is added to the root. If the tree is not balanced, the shallowest rightmost node where the insertion does not increase the height of the tree is selected. These two cases are also illustrated in Figure 2b and Figure 2c , where Figure 2b contains the case of an unbalanced tree and Figure 2c contains the case of a balanced tree. Updating the keys in the tree is

now handled by the rightmost leaf in the subtree rooted at the insertion node, which is also called the sponsor. For this, the sponsor calculates the needed keys k and blinded keys BK and broadcasts the calculated BKs to the new member and the old members of the group. Thus, all the old group members and the new member have all the keys they need to calculate the group key k_{group} for the updated group.

(3) Member Revocation: Removing a member starts with the member to be revoked broadcasting a leave request containing its blinded key BK to the other group members. Then the remaining group members update their key tree structures by removing the revoked member node and replacing its former parent node with the former sibling node of the node to be removed. Such a tree update for an example removal process is also illustrated in Figure 3. Updating the keys in the new key tree is again done by the so-called sponsor. In this case the sponsor is the right most node rooted at the leaving members sibling node. Analogous to adding a new group member, the sponsor computes for itself all keys and blinded keys for the new tree structure and broadcasts the blinded keys to the remaining group members. In this way all group members have again all needed information for the calculation of the group key.

III. Evaluation Environment

Now that we have shown how the distributed scheme works, we present the evaluation environment in which we evaluate the performance of the scheme. For this purpose, we present (1) a measurement setup, (2) metrics, and (3) workloads for the distributed group encrypion scheme.

A. Measurement Setup

Analogously to our previous papers [2], [3], [5], [6] and [7] we use a EPS microcontroller as the hardware for the realization of an IoT scenario. More specific, we use the ESP8266, a widely used microcontroller in various IoT scenarios such as data centre temperature monitoring [8], smart home automation [9] or garbage level monitoring in big cities [10]. The ESP8266 is a so-called System-on-a-Chip and a 32-bit microcontroller from Espressif Systems. It has the ability to establish a 2.4 GHz Wi-Fi connection, a 80 MHz dual-core CPU and 64 kB RAM. A ESP8266 represents one group member and for our measurements we used up to 5 ESP8266s, as shown in Figure 4. Also shown in Figure 4. is a Raspberry Pi 3B+ running an MQTT broker that allows the ESPs to communicate with each other. As software for the realization of the MQTT broker we used the Mosquitto Broker in version 1.5.7. The reason why we decided to have the ESPs communicate via an MQTT broker is that MQTT is one commonly applied protocol for the realization of IoT communication [11]. However, in order for the ESPs to communicate via the broker, they themselves require an MQTT client implementation, for which we use the Async MQTT client [12]. In addition, we use on the ESPs the gmp-ino library [13], a port of the gmp library, to perform the calculations necessary for the distributed group encryption scheme.

B. Workload Pattern

In terms of workloads, we focus on the group management operations that provide the corresponding keys for encrypting the group communication. The reason why we do not consider group communication as a workload is that group encryption schemes are typically used to distribute a key that is then used in combination with an additional and significantly faster symmetric encryption scheme such as AES. Therefore, we do not consider the encryption and decryption of the actual group communication as a workload of the distributed group encryption method, but only the generation of the group key. Therefore, we consider the addition and removal of group members as workload and define them as follows. Thereby we distinguish between sponsor and non-sponsor.

(1) Addition and (2) Revocation of a Group Member: The sponsor/non-sponsor group members of a group with m members performs the necessary calculations to update its key trees and group key after a member is added or removed from this group n times in a row.

C. Metrics

In order to evaluate the performance of the distributed group encryption scheme, we consider as metrics the time required to perform group management operations and the storage space required. Regarding times, we consider the average time it takes to (i) add a member to the group t_a, and (ii) revoke a member from the group t_r. Thus t_a can be exemplary calculated using Equation 3, where n stands for the number of addition operations performed and t_{a_i} for the time of the i-th addition operation. As accuracy, we use the standard deviation of the measured quantities. Analogously t_r can be calculated.

$$t_a = \frac{1}{n} \sum_{i=1}^{n} t_{a_i} \tag{3}$$

In addition to the required times, we also take into account the average memory requirements. Specifically, we consider the average memory requirements for storing the temporarily required information for updating the key tree and recalculating the group key. As a measure of accuracy we again use the standard deviation of the measured sizes as an error.

IV. Performnance Evaluation

In this section, we first present the performance analysis of distributed scheme [4] in our IoT testbed and then compare it centralized schemes.

A. Performance Analysis of the Scheme [4] in an IoT Use Case

We start the performance analysis of the distributed system on the basis of adding members. To do this, we first consider the time it takes the sponsor to update the key tree to add a new member to the group. The times required by the sponsor to update the key tree can be seen in Figure 5, showing that (i) the required time depends on the height of the sponsor in the tree and (ii) the greater the sponsor's height in the tree, the

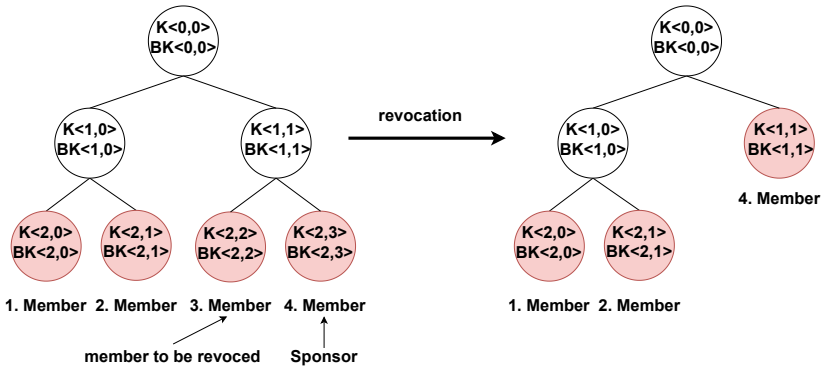

Fig. 3. Illustration of the revocation of a member from a group with four members

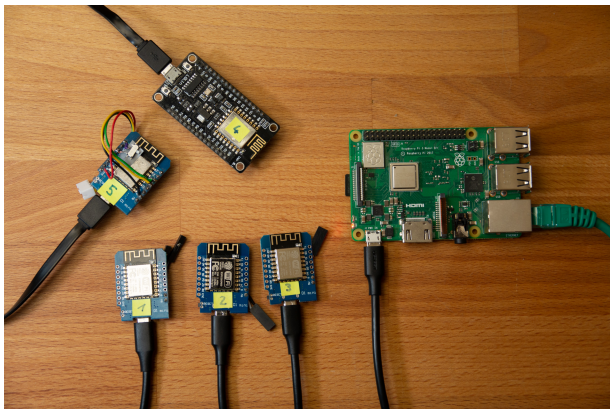

Fig. 4. Measurement setup for up to 5 group members, each realized using an ESP8266

greater the time required. After the sponsor updates the key tree, it broadcasts it to the other group members via MQTT so that they can calculate the new group key from the updated key tree.

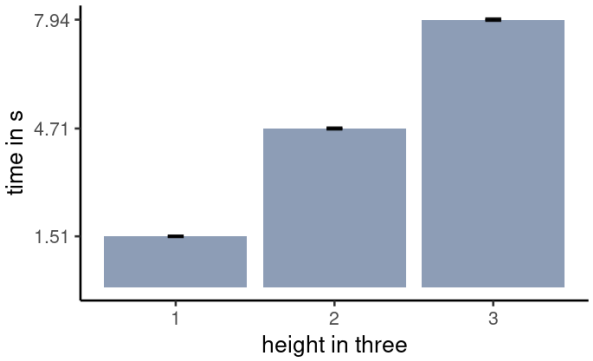

Fig. 5. Time needed by the sponsor to update the key tree, depending on the height of the sponsor in the tree

The total time required to add a new group member, i.e., the time required for the sponsor to compute the new key tree, the subsequent broadcasting of the new tree, and the computation of the new group key by all group members is illustrated in Figure 6. More precisely, Figure 6 shows the times needed to add the second, third, fourth and fifth member to a tree with initial only one member. Thereby it can be seen that the time needed increases with the number of already existing members.

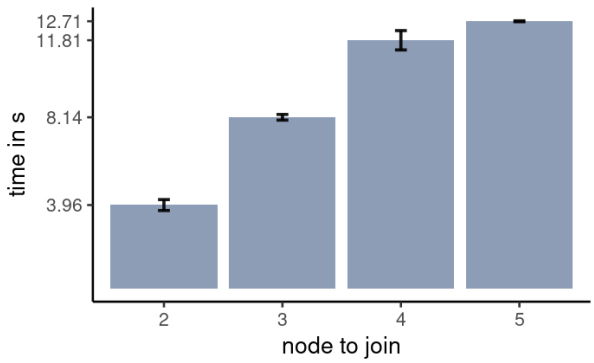

Fig. 6. Total time required to add a new group member, i.e., the time required for the sponsor to compute the new key tree, the subsequent broadcasting of the new tree, and the computation of the new group key by all group members

The time required to remove a member is almost identical for adding a member. The only difference is how the sponsor has to modify the underlying tree structure differently in order to subsequently compute the new key tree. However, the time required for this is so marginal that we could not measure it within our measurement accuracy. Thus, the time required by the sponsor again depends only on its height in the tree and corresponds to the times for adding members shown in Figure 5. Analogously, the total time required to remove a member is the same as the total time required to add a member in Figure 6.

Finally, we consider the memory requirements for temporarily storing the new key tree generated by the sponsor. We have measured this as an example for the case when 4 new members are successively added to an initial group consisting of only one member. The size of the key tree, which is generated and broadcast by the sponsor after each addition, can be seen in Figure 7. In this figure, we can see that the memory requirements of the key tree increase linearly with the number

of existing members in the group.

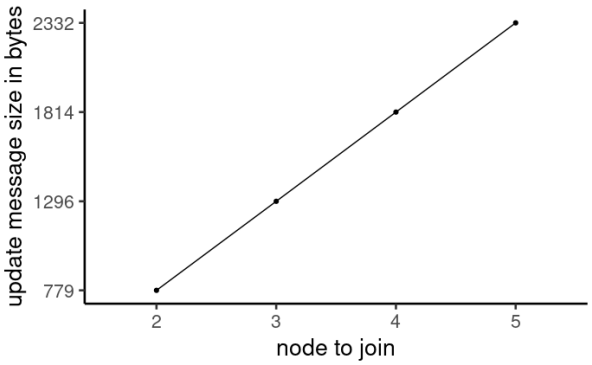

Fig. 7. Memory requirements of the key tree, which the sponsor broadcasts after sequentially adding four new members in an initial group with only one member

B. Performance Comparison of the Distributed Scheme [4] with Centralized Schemes

In order to better classify the previously presented performance of the distributed system, we now compare it with the performance of centralized procedures. We use as comparison schemes the centralized schemes $SKDC$ and the scheme proposed by Nishat et al., whose performance we have already analyzed in [3] and [2]. The performance of these two schemes was measured once on an ESP32, which acted as hardware for a group member, and on a commercially available laptop, which acted as the central instance.

The performance measurements showed that in case of the scheme proposed by Nishat et al., that the group members had to temporarily store almost 20 Bytes per existing group member. If we compare these values with the memory requirements of the distributed scheme [4], we see that Nishat could manage 38, 64, 90 and 116 members respectively with the memory required by the distributed scheme for 2,3,4 and 5 members respectively. In case of $SKDC$, on the other hand, the group members had to temporarily store only 16 Bytes, regardless of the group size. Thus, two statements can be made with regard to the temporary memory requirements. First, the two centralized schemes require significantly less temporary memory than the distributed scheme. Second, the centralized scheme $SKDC$ supports, in contrast to the distributed scheme [4] and scheme proposed by Nishat et al., arbitrarily large groups in terms of memory requirements. Nevertheless, it should not be forgotten at this point that the distributed scheme is generally applicable to IoT and has been shown to support up to 5 members in terms of temporary memory requirements (although even more members would have been possible). In terms of the platooning use case described at the beginning, this would mean that the distributed scheme would support at least 5 cars driving behind each other in a platoon.

Finally, we consider the time required for a group member to join an existing group. In the distributed method, it takes 3.96 seconds, 8.14 seconds, 11.81 seconds and 12.71 seconds for the second, third, fourth, and fifth members to join the group, respectively, see Figure 6. The centralized schemes $SKDC$ and the scheme proposed by Nishat et al. schemes, on the other hand, required time under 3 seconds for up to 1000 members. (Note: Since these statements are based on measurements using the ESP32 and not the ESP8266, we repeated the measurements on the ESP8266. These measurements confirmed that the statements also have validity for the case when the ESP8266 is used as hardware for the group members and not the ESP32). Thus, the statement that the centralized schemes are significantly better than the distributed scheme is also valid with regard to the calculation times. However, three facts must not be forgotten here. First, the centralized schemes require a trusted third party, also called a central authority, which takes on the main burden of the calculations for them. However, this also means that the group members must be able to communicate with this central instance. This may not always be the case, as in the platooning example, when the vehicles that want to form a platoon drive through a tunnel or through an area with poor Internet connection. Second, all the measurements performed were under perfect network conditions, with the devices in close proximity to each other, see e.g. Figure 4. This scenario is a more realistic assumption for the distributed scheme in the platooning case, where cars drive close to each other, than in the case of centralized schemes, where the central instance can be located far away in a cloud. Accordingly, in centralized schemes, network effects and transmission times can still significantly increase the required times. And third, the distributed scheme allows 5 members to form a group in under 40 seconds, which is an acceptable duration in the case of a platoon, for example, in which cars want to ride close together in a group over long distances.

In summary, our performance analyses show that, under optimal network conditions, the distributed scheme [4] performs worse than the centralized schemes $SKDC$ and $Nishat$ in terms of required times and memory requirements. Nevertheless, the distributed scheme allows the creation of small groups as long as the use case allows the creation time to be in the seconds or minutes range. However, the distributed scheme does not require a central instance for this and is less dependent on network effects. Therefore, we plan to extend our work in the future to include a performance analysis of the schemes in different network situations.

V. Related Work

In this section, we review related work and highlight the novelty of our contribution. The literature regarding the evaluation of the performance of encryption schemes in IoT use cases can be divided into the following four areas: (1) 1-to-1 encryption, (2) surveys, (3) centralized schemes, and (4) distributed procedures. For example, in [14], [15] and [16], the authors analyze the performance of 1-to- 1 encryption schemes in an IoT environment. Besides these practical performance analysis of 1-to-1 encryption schemes in the IoT environment, practical performance analysis of group or n-to-n encryption schemes in the IoT environment also exist in the literature.

These mainly focus on centralized schemes, as we have also done in [2] and [3], for example. In addition to conventional group encryption schemes, attribute-based encryption schemes have also been evaluated on IoT devices ([17], [18] and [19]) , which are in principle also centralized group encryption schemes, except that they additionally supports subgroups. We differ from these works in that we analyze the practical performance of decentralized group encryption schemes on IoT devices.

However, the performance of distributed schemes are also already considered in the literature. For example, in the form of theoretical performance analyses in surveys such as [20] and [21], but also in practical performance analysis such as [22]. These performance analysis are either theoretical estimations using the Landau Notation or were carried out on much more powerful hardware than is usual used in the IoT environment. Like in [22], where the authors evaluated the performance of the distributed scheme proposed by Kim et al. on 666 MHz Pentium III dual-processor PCs running Linux and communicating via the Linux inter-process communication pipe fifo. In principle, we performed a similar performance analysis for the distributed scheme proposed by Kim et al. but in an IoT use case. Unlike [22], our contribution thus allows us to evaluate the performance of the Kim et al. scheme on IoT devices, that use IoT typical communication protocols. This provides important insights for developers looking for a suitable scheme for an IoT use case or for researchers looking at the performance of distributed schemes on IoT hardware. For example, adding or removing a group member using the distributed scheme [4] on IoT devices already takes more than 1 second and up to almost 13 seconds for group sizes below 6 members, whereas in [22] it was measured that adding and removing members only takes under 0.5 seconds for group sizes up to 50 members. In addition, unlike [22], we also compare the performance of the distributed scheme proposed by Kim et al. with centralized schemes.

VI. Conclusion

To ensure secure communication between IoT devices, a centralized or distributed group encryption scheme can be used. However, both of these approaches have not yet been developed for IoT devices, and thus their performance on such devices is unknown. In this work, we investigate the performance of the distributed group encryption scheme proposed by Kim et al. on IoT devices, analogous to our previous work [2], [3] in which we investigated the performance of centralized group encryption schemes on IoT devices. To this end, we build a measurement environment for the distributed group encryption scheme, analyze the performance of the distributed scheme, and compare the performance of centralized and distributed group encryption schemes in an IoT scenario. Our results show that the distributed encryption scheme of Kim et al. performs worse than the centralized scheme SKDC and the centralized scheme proposed by Nishat et al. under perfect network conditions in terms of time and memory requirements. However, the distributed method allows a group consisting of 5 members to agree on a group key in under a minute. Thus, while our measurements proved that the distributed scheme, under perfect network conditions, performs worse than the centralized schemes, they also proved that the distributed scheme can be used for non-time-critical IoT use cases.

However, it should not be forgotten that the distributed scheme is less dependent on network effects and thus it could also perform better than the centralized schemes under poor network conditions. To clarify this, we plan to compare the performance of centralized and distributed group encryption schemes in limited network conditions in future work our benchmark framework [23] for Publish/Subscribe communication protocols under network limitations and our IoT network emulator [24].

Further, we plan to integrate situation-awareness. Such a situation-aware mechanism can switch the encryption scheme based on the current system situation based on self-learning decision-making, e.g., following a self-aware computing systems approach [25].

Acknowledgment

This research has been funded by the Federal Ministry of Education and Research of Germany in the framework KMU-innovativ - Verbundprojekt: Secure Internet of Things Management Platform - SIMPL (project number 16KIS0852) [26].

References

[1] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, vol. 22, pp. 97–114, 2009.

[2] T. Prantl *et al.*, "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)*, 2020, pp. 1–8.

[3] ——, "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*, ser. ICPE '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 233–240.

[4] Y. Kim *et al.*, "Tree-based group key agreement," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, p. 60–96, feb 2004.

[5] T. Prantl *et al.*, "Performance evaluation of a post-quantum public-key cryptosystem," in *2021 IEEE 40th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2021.

[6] T. Prantl, T. Zeck, L. Iffländer, L. Beierlieb, A. Dmitrenko, C. Krupitzer, and S. Kounev, "Towards a cryptography benchmark: A view on attribute based encryption schemes," in *2022 5th Conference on Cloud and Internet of Things (CIoT)*. IEEE, 2022.

[7] T. Prantl *et al.*, "Benchmarking of pre- and post-quantum group encryption schemes with focus on iot," in *IPCCC*, 2021.

[8] S. Saha *et al.*, "Data centre temperature monitoring with esp8266 based wireless sensor network and cloud based dashboard with real time alert system," in *2017 Devices for Integrated Circuit (DevIC)*, 2017, pp. 307–310.

[9] R. K. Kodali *et al.*, "Mqtt based home automation system using esp8266," in *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, 2016.

[10] J. D. Kothari, "Garbage level monitoring device using internet of things with esp8266," *Jubin Dipakkumar Kothari (2018). Garbage Level Monitoring Device Using Internet of Things with ESP8266. International Journal of Innovative Research in Computer and Communication Engineering*, vol. 7, no. 6, pp. 2995–2998, 2018.

[11] G. Perrone *et al.*, "The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices," in *2nd IoTBDS*, 01 2017, pp. 246–253.

[12] S. Merlin *et al.*, "Async MQTT client for ESP8266 and ESP32," 2018, online available under https://github.com/marvinroger/async-mqtt-client, Accessed on 30.12.2021.
[13] C. A. Ferraris, "gmp-ino," 2020, online available under https://github.com/CAFxX/gmp-ino, Accessed on 30.12.2021.
[14] A. Braeken *et al.*, "Anonymous lightweight proxy based key agreement for iot (alpka)," *Wireless Personal Communications*, vol. 106, no. 2, pp. 345–364, 2019.
[15] A. K. Das *et al.*, "Provably secure ecc-based device access control and key agreement protocol for iot environment," *IEEE Access*, vol. 7, pp. 55 382–55 397, 2019.
[16] T. Prantl *et al.*, *Performance Impact Analysis of Securing MQTT Using TLS*. New York, NY, USA: Association for Computing Machinery, 2021, p. 241–248.
[17] M. Fischer *et al.*, "Using attribute-based encryption on iot devices with instant key revocation," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2019, pp. 126–131.
[18] N. Oualha *et al.*, "Lightweight attribute-based encryption for the internet of things," in *2016 25th ICCCN*. IEEE, 2016, pp. 1–6.
[19] L. Touati *et al.*, "C-cp-abe: Cooperative ciphertext policy attribute-based encryption for the internet of things," in *2014 INDS*. IEEE, 2014.
[20] R. Dutta and R. Barua, "Overview of key agreement protocols." *IACR Cryptol. ePrint Arch.*, vol. 2005, p. 289, 2005.
[21] J. V. D. Merwe *et al.*, "A survey on peer-to-peer key management for mobile ad hoc networks," *ACM Comput. Surv.*, vol. 39, no. 1, p. 1–es, apr 2007.
[22] Y. Amir *et al.*, "On the performance of group key agreement protocols," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 3, p. 457–488, aug 2004.
[23] S. Herrnleben *et al.*, "Combench: A benchmarking framework for publish/subscribe communication protocols under network limitations," in *EAI International Conference on Performance Evaluation Methodologies and Tools*. Springer, 2021.
[24] ——, "An IoT Network Emulator for Analyzing the Influence of Varying Network Quality," in *Simulation Tools and Techniques*. Cham: Springer International Publishing, 2021, pp. 580–599.
[25] C. Krupitzer *et al.*, "An Overview of Design Patterns for Self-Adaptive Systems in the Context of the Internet of Things," *IEEE Access*, vol. 8, pp. 187 384–187 399, 2020.
[26] T. Prantl *et al.*, "Simpl: Secure iot management platform," in *ITG Workshop on IT Security (ITSec)*, 2020.