Evaluating the Performance of a State-of-the-Art Group-oriented Encryption Scheme for Dynamic Groups in an IoT Scenario

Thomas Prantl, Peter Ten, Lukas Iffländer, Alexandra Dmitrenko, Samuel Kounev, Christian Krupitzer
University of Würzburg
{thomas.prantl|peter.ten|lukas.iffländer|alexandra.dmitrenko|samuel.kounev|christian.krupitzer}@uni-wuerzburg.de

***Abstract*—New emerging technologies, such as autonomous driving, intelligent buildings, and smart cities, are promising to revolutionize user experience and offer new services. The world has to undergo large scale deployment of billions of things — cost-efficient intelligent sensors that will be interconnected into extensive networks and will collect and supply data to intelligent algorithms — to make it happen. To date, however, it is challenging to secure such an infrastructure for many-fold reasons, such as resource constraints of things, large scale deployment, many-to-many communication patterns, and dynamically changing communication groups. All these factors rule out most of the state-of-the-art encryption and key-management techniques.**

Group encryption algorithms are well-suitable for many-to-many communication patterns typical for IoT networks, and many of them can deal with dynamic groups. There are, however, very few constructions that could potentially fulfill the computational and storage constraints of IoT devices while providing sufficient scalability for large networks. The promising candidates, such as construction by Nishat et al. [1], were not evaluated using IoT platforms and under constraints typical for IoT networks.

In this paper, we aim to fill this gap and present the evaluation of a state-of-the-art group-oriented encryption scheme by Nishat et al. to identify its applicability to IoT systems. In detail, we provide a measurement workflow, a revised version of the approach, and describe a reproducible hardware testbed. Using this evaluation environment, we analyze the performance of the encryption scheme in a typical IoT scenario from a group member perspective. The results show that all calculation times can be assumed to be constant and are always below 2 seconds. The memory requirement for permanent parameters can also be considered to be constant and are below 8.5 kbit in each case. However, the information that has to be stored temporarily for group updates has turned out to be the bottleneck of the scheme, since their memory requirements increase linearly with the group size.

***Index Terms*—Security, Group Encryption, Performance Evaluation, Internet of Things**

I. Introduction

In today's era of digital transformation, we observe the synergy of physical and digital worlds through the integration of billions of things — cost-effective Internet-of-Things (IoT) platforms — into physical objects. This synergy has the potential to improve our daily environment through data collection, analysis, and automation of routine tasks and laying the foundations for additional intelligence in many use cases, such as smart factories, smart cities, smart homes, or smart health. The additional "smartness" comes from many IoT devices collecting and sharing data, and intelligent algorithms processing it.

The number of deployed IoT devices continuously increases, and — according to the estimation of a Gartner report [2] — this year, there will be 20 billion IoT devices connected to the Internet. The amount of data generated and shared by these devices increases immeasurably, considering that all connected IoT devices in 2018 have already generated five quintillion bytes of data [3]. However, not only the pure volume of data increases but also the complexity of communication patterns, which typically involve groups of devices interested in sharing data with constantly changing group membership dynamics. For instance, various types of wearable devices can monitor the health state of humans and send the collected data to patients, health insurance, or doctors [4]. Thereby, it is essential that the users are always in control of their data and can determine who can read it. Unauthorized control to transmitted data can be achieved through encryption so that only authorized parties that have a corresponding decryption key can decrypt it. As the amount of shared data with different groups of people will grow enormously in the future, it will become more and more important that the corresponding encryption methods work as efficiently as possible. Especially in the context of IoT systems, where IoT devices are limited in storage capacity, have little computing power, and often have only a limited power supply, efficiency becomes a key factor.

Many group encryption methods emerged requiring only one encryption for all the (e.g. [5], [6]) to boost efficiency in systems that feature one-to-many or many-to-many communication patterns. These schemes enable the efficient encryption of messages for fixed groups but become costly when modifying the group's members. The necessary modifications caused by group membership changes often affect not only a single group member but several or even all group members [1]. Accordingly, for scenarios with dynamic groups, such as IoT, these modifications must be as efficient as possible, since they may occur very frequently. A new promising method was presented in [1] to deal efficiently with dynamic groups, which enables efficient group changes with constant overhead, in contrast to existing schemes [1] that typically feature more significant overhead (linear or logarithmic).

In this work, we complement [1] through an evaluation of the approach in an IoT setting with real devices. In [1], the authors focused on computation time. We further analyze the efficiency of the scheme concerning energy consumption, energy efficiency, and data overhead. Especially for the last aspect, we focus on real values in terms of bytes rather than providing estimation using the Landau notation, as done on the original publication.

Specifically, this paper provides the following contributions:

- the description of a workflow, how to use the scheme in practice,
- a revised version of the approach from [1] eliminating an error in determining the sent information size in case of group changes,
- the design of a reproducible hardware testbed for the performance measurements of the scheme in an IoT scenario,
- the definition of performance metrics, including error measures, to rate the performance of the scheme, and
- an analysis of the general performance of the scheme using our testbed.

The remainder of this paper is structured as follows. In Section II, we present the scheme [1] in detail. In Section III, we highlight an error in the original publication, which affects the amount of transferd information in case of group changes, and propose a fix along with a correct estimation. Section IV introduces our practical performance evaluation environment containing the measurement setup, workload patterns, and metrics. Section V deals with the proposed practical performance evaluation and reports results.

II. Background

In the course of this section, we introduce the most important components and processes of the scheme developed by Nishat et al. [1]. Specifically, we present (i) the involved actors, including their tasks and connections, (ii) the initial creation of a group consisting of n members, (iii) the addition and (iv) the revocation of members from this group, and (v) the encryption and decryption of messages by group members. Figure 1 illustrates the approach.

As we already mentioned, the original paper has an error, which we will explain in more detail in Section III. Here, we use the corrected notation. We also describe a procedure for the use of the scheme, which includes practical recommendations such as updating parameters only once after a group change.

(1) Involved Actors: Those consist of the actual group members, who should be able to communicate securely with each other, and a central instance (CI), which is responsible for creating and managing groups. In general, each member of the group could take over this rule, but in the following, we assume a dedicated CI. In a smart home scenario, for example, the corresponding smart devices could form a group, and the home server could act as the CI. Since the CI manages the group memberships, it generates the corresponding keys and parameters for decryption and encryption and distributes them to the group members and can read all group messages. Thus, the CI must be a trustworthy party. Secure communication must be available once between the CI and each group member for the inclusion in a group. For subsequent communication between a group member and the CI, an insecure connection suffices as long as it guarantees all messages reaching their destination without artificial delay. Otherwise, it could happen that, e.g., when removing a member, another member misses this updated and continues to encrypt messages so that the excluded member can read them. For the rest of this paper, we assume that no messages are delayed or intercepted. For the sake of simplicity, we assume an integrated group management approach, i.e., the CI knows all initial group members, and when to add or remove members.

(2) Initial Group Creation: Figure 1a shows the initial group creation, which requires the CI to determine the master secret key MSK, the system parameter Γ and mathematical implementation details I. These parameters are used by the CI to determine the corresponding encryption and decryption information (namely the public key PK and respectively the public parameter γ and the secret keys S_i) used by the group members.

For creating a new group, the CI first determines a prime number p, to generate two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$, each of order p. Thereby, g is the generator of G_1. Then the CI randomly chooses $\alpha, \beta \in \mathbb{Z}_p^*$ and $h \in \mathbb{G}_1$. These values already determine the $MSK = (\alpha, \beta)$ and the system parameter $\Gamma = (h, g, g^\alpha, g^\beta)$. Next, the CI determines the public key PK and the secret keys of the $n \in \mathbb{N}$ initial group members. For this purpose, the CI first assigns each initial group member a prime number $r_i \in \mathbb{Z}_p^*$. Thereby, r_i is the assigned prime number of the i-th group member. These prime numbers are now used by the CI to select randomly $k \in \mathbb{Z}_p^*$, such that $k < min\{r_1, ..., r_n\}$. Using the above-introduced parameters and Equations 1 and 2, the CI can now calculate PK and the secret keys of the initial group members. Equation 2 shows the calculation of the secret key S_i of the i-th user.

$$PK = (\underbrace{g^{\alpha k}}_{PK_1}, \underbrace{g^{\beta k}}_{PK_2}) \tag{1}$$

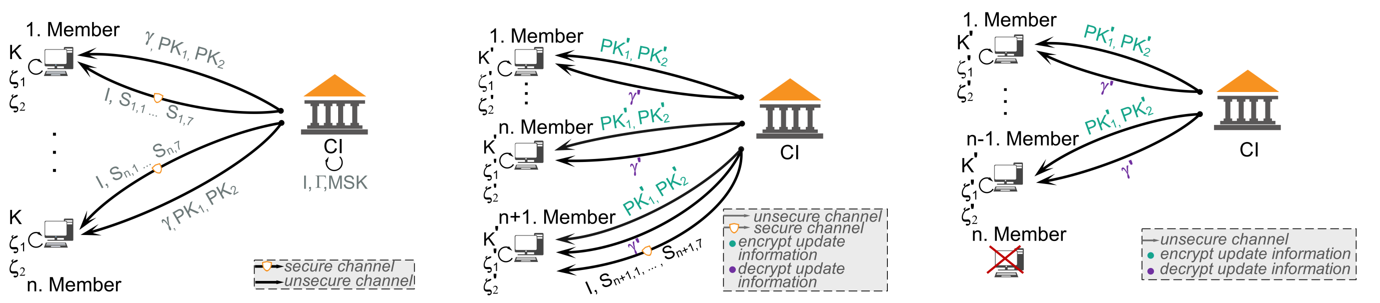
$$S_i = (\underbrace{h^{r_i}}_{s_{i_1}}, \underbrace{g^{r_i}}_{s_{i_2}}, \underbrace{h^{\frac{r_i}{\beta}}}_{s_{i_3}}, \underbrace{g^{\frac{r_i \alpha}{\beta}}}_{s_{i_4}}, \underbrace{h^{\frac{1}{\beta}}}_{s_{i_5}}, \underbrace{gh^{r_i}}_{s_{i_6}}, \underbrace{r_i}_{s_{i_7}}) \tag{2}$$

The public parameter γ results from using equation 3, for which $a \in \mathbb{Z}_p^*$ must be chosen randomly.

$$\gamma = (a * r_1 * ... * r_n) - k \tag{3}$$

In accordance with [1], a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ must also be defined by the CI, which is used by all group members for later encryption and decryption. The bilinear map e and the parameters g and h form the mathematical implementation information I. Thereby, I is required by the group members for encryption and decryption.

CI must send the members their respective secret key S_i and the mathematical implementation details I via a secure



(a) Group creation (b) Member Addition (c) Member Revocation

Fig. 1: Visualization of the group management operations (a) group creation, (b) member addition and (c) member revocation (whereby it is assumed, that the CI knows all initial group members and when and which member it should add or remove)

channel to complete the group creation. The CI must also send γ and PK to all group members, but this does not require a secure channel. Using Equitation 4, 5 and 6, the individual group members can then calculate k, ζ_{i_1}, and ζ_{i_2}, which they require for decryption.

$$\begin{aligned} k &= (r_i - \gamma) \bmod r_i && (4) \\ \zeta_{i_1} &= s_{i_1} s_{i_2}^k && (5) \\ \zeta_{i_2} &= \frac{s_{i_3} s_{i_4}^k}{s_{i_5}^{k-1}} && (6) \end{aligned}$$

(3) Addition of Group Members: As shown in Figure 1b, adding group members requires the CI to provide a secret key S_{n+1} for the new member and to update the PK and γ for all group members, so that they can recalculate k, ζ_{i_1}, and ζ_{i_2}. Thereby, updating PK is necessary to encrypt messages to the new group while updating γ is necessary to decrypt messages in the new group. S_{n+1} is calculated analogously as for the old group members using Equation 2, which requires the selection of a random prime number $r_{n+1} \in \mathbb{Z}_p^*$. The parameters γ and PK are updated according to Equation 7 and Equation 8, wherefore a new random $a' \in \mathbb{Z}_p^*$ and $k' \in \mathbb{Z}_p^*$, where $k' < min\{r_1, ..., r_n, r_{n+1}\}$, must be selected.

$$\begin{aligned} \gamma' &= (\gamma + k) * a^{-1} * a' * r_I - k' && (7) \\ PK &= (\underbrace{g^{\alpha k'}}_{PK_1'}, \underbrace{g^{\beta k'}}_{PK_2'}) && (8) \end{aligned}$$

(4) Revocation of Group Members: As shown in Figure 1c, revoking group members requires the CI to update both the PK and γ, so that all remaining members can recalculate k, ζ_{i_1}, and ζ_{i_2}. This is done by means of Equation 9 and Equation 10, for which a random $a' \in \mathbb{Z}_p^*$ and $k' \in \mathbb{Z}_p^*$, where k' is smaller than each r_i of the remaining members, must be selected. The factor r_R in Equation 9 is the assigned prime number of the group member who should be excluded.

$$\begin{aligned} \gamma' &= (\gamma + k) * a^{-1} * a' * (r_R)^{-1} - k' && (9) \\ PK &= (\underbrace{g^{\alpha k'}}_{PK_1}, \underbrace{g^{\beta k'}}_{PK_2}) && (10) \end{aligned}$$

(5) Encryption and Decryption of a Message: Encrypting a message M or decrypting a ciphertext C is possible for every group member based on Equation 11 or Equation 12. Encryption of messages additionally requires random selection of $t \in \mathbb{Z}_p^*$ by the corresponding group member.

$$C = (\underbrace{M * e(g, PK_1)^t}_{c_1}, \underbrace{(h * PK_1)^t}_{c_2}, \underbrace{PK_2^t}_{c_3}) \quad (11)$$

$$M = \frac{c_1 e(\zeta_{i_1}, c_2)}{e(s_{i_6}, c_2) e(\zeta_{i_2}, c_3)} \quad (12)$$

III. Theoretical Evaluation

While implementing the procedure proposed by Nishat et al. [1], we noticed a small notation error in the calculation of the public parameter. In this section, we show (i) this error by presenting a counterexample, (ii) how this error can be corrected, and (iii) how it changes the theoretical performance of the method.

The error in the original publication [1] occurs when calculating γ, which each member uses to calculate back k. Equation 13 shows the original equation from [1]. Consider the following counterexample to illustrate the error: We choose p as 13, a as 3, r_1 as 5, r_2 as 7, and k as 2. In this case, the CI would calculate γ as 12 (with accordance to the Equation 14) and distribute it to the members. If the first group member tries to calculate k, he gets the value 3 (cf. Equation 15), instead of 2. Thus the transmission of k by sending γ to the group members did not work.

$$\begin{aligned} \gamma &= a \times r_1 ... \times r_n - k && (13) \\ \gamma &= (3 * 5 * 7 \bmod 13 - 2) \bmod 13 = 12 && (14) \\ k &= (5 - 12 \bmod 5) \bmod 13 = 3 && (15) \end{aligned}$$

This error originates from Equation 1 in the original publication, which assumes that $(a \times r_1 \times ... \times r_n \bmod r_i)$

equals 0, which is only the case when using the "ordinary" multiplication $*$ and not the group operation $\times$. Therefore, it should be possible to correct this error by using $*$ instead of $\times$.

In the original publication, the authors estimated that the size of $\gamma \in \mathcal{O}(1)$, since $\gamma \in \mathbb{Z}_p^*$. To correct this estimation, we investigate the corrected Equation 3. For this equation, we first analyze the size of the minuend and subtrahend individually. Since the subtrahend $\in \mathbb{Z}_p^*$ its size is $\in \mathcal{O}(1)$. The minuend is the product of $n+1$ positive numbers, all smaller than p, whereby n is the number of group members. Therefore, the size of the minuend is $\in \mathcal{O}(n)$. Thus, the size of γ is $\in \mathcal{O}(n)$ in total.

The remaining theoretical estimates of the original paper are not affected by the changes made, as for them only the size of the affected parameter changes, but not the general approach. Nevertheless, we refuted the claim that the scheme presented in [1] only has fixed costs, since the size of γ grows linearly with the group size.

IV. Proposed Evaluation Environment

As not all costs are theoretically constant anymore as claimed by the original publication [1] — because the size of γ is $\in \mathcal{O}(n)$ — it is even more critical to determine the costs for an application of the approach in practice. The scheme [1] no longer supports arbitrarily large groups — since the group creation requires the temporary storing of γ, which grows linearly with the group size. However, it could still support group sizes that may be sufficiently large for many practical applications in the IoT context. For this reason, we analyze the performance of the scheme in a suitable environment. The procedure focuses on large groups perfectly suiting the requirements of IoT systems with increasing group sizes. Hence, we carry out our analysis in an IoT typical environment. Thereby, we assume that small low-power IoT devices form the members of a group, and a powerful PC or server acts as the CI. In terms of performance analysis, we focus on four specific aspects relevant in the context of resource-scarce IoT devices: storage requirements, computation time, power consumption, and energy efficiency. Throughout this section, we introduce (i) the measurement setup used for performance measurements, (ii) the used workload patterns, and (iii) the applied metrics to evaluate performance.

A. Workload Pattern

Since we are interested in the performance of the scheme on a typical IoT device, all workload patterns describe the behavior of an IoT device. In the following, we describe four different workload patterns, which describe the different scenarios of an IoT device we take into account for evaluation.

(1) Encryption and *(2) Decryption:* The IoT device encrypts/decrypts a message consisting of B bytes N times in a row. Thus, a total of $B*N$ bytes are encrypted/decrypted.

(3) Updating Decryption Information: The IoT device recalculates the parameters k, ζ_1, and ζ_2 from a stored fixed γ in total N times.

(4) IDLE: The IoT device is running in an idle state.

B. Metrics

Since IoT devices typically have not only limited hardware resources but also limited power supply, it is especially critical to use the available energy as efficiently as possible. Therefore, we use power consumption and energy efficiency, among other things, as comparative metrics to evaluate the measurement results.

For power consumption, we introduce the abbreviation W and define it in Equation 16 as the average power consumption per second. In this equation, n stands for the duration of the measurement in seconds and W_i for the power consumption during the ith second. In Equitation 17, we determine the accuracy of W using Gaussian error propagation, to be able to indicate an error range in which the actual value of W is located. Thereby ΔW_i is the accuracy of the measured power consumption during the i-th second. All power measurements use a Yokogawa WT310, whose power measurement error is $\pm$(0.1% of reading + 0.2% of range), according to the manufacturer [7]. The range error in our case is 0.0006 Watt because we have set the measuring ranges to 3V and 100mA. These considerations result in the final calculation of ΔW according to Equation 18.

$$W = \frac{1}{n}\sum_{i=1}^{n} W_i \tag{16}$$

$$\Delta W = \frac{1}{n}\sqrt{\sum_{i=1}^{n} \Delta W_i^2} \tag{17}$$

$$\Delta W = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(0.1\% * W_i + 0.0006 * W)^2} \tag{18}$$

Under SPEC specifications [8], we define in Equation 19 the energy efficiency E as the ratio of the throughput T (see Equation 21) to the power consumption W. We calculate the accuracy of the energy efficiency again using Gaussian error propagation, see Equation 20.

$$E = \frac{\text{Throughput}}{\text{Power Consumption}} = \frac{T}{W} \tag{19}$$

$$\Delta E = \sqrt{\frac{\Delta T^2}{W^2} + \frac{T^2 * \Delta W^2}{W^4}} \tag{20}$$

As throughput, we consider the operations performed during a specified period t. We distinguish several operations: (i) Encrypting B_e bytes, (ii) decrypting B_d bytes. For example, the throughput for the encryption of messages alone results from Equation 21. The throughput accuracy is obtained by Gaussian error propagation according to Equation 22, whereby we assume that each encryption process has been successful. Thereby, Δt is the accuracy of the observed time frame t.

$$T_e = \frac{B_e}{t} \tag{21}$$

$$\Delta T_e = \frac{B_e * \Delta t}{t^2} \tag{22}$$

The throughput for decryption can be calculated analogously by exchanging the index of B_e accordingly. A third operation, not yet mentioned, is to perform group update operations. The corresponding throughput can also be calculated analogously but in this case B_e must be replaced by the number of the performed group update operations N_u. Since the update process for the encryption information consists only of storing PK, we only consider the update process of the decryption information for group updates here and in the following.

In addition to energy efficiency and power consumption, we also consider as metrics the average time it takes to (i) encrypt messages t_e, (ii) decrypt messages t_d, and (iii) perform group update operations t_u. t_e can be calculated using Equation 23, where n stands for the number of encryption operations performed and t_{e_i} for the time of the i-th encryption operation. The error can be determined by Gaussian error propagation using Equation 24. Here, t_{e_i} stands for the accuracy of the determination of the time duration for the i-th encryption process. t_d and t_u can be calculated analogously.

$$t_e = \frac{1}{n}\sum_{i=1}^{n} t_{e_i} \qquad (23) \qquad\qquad \Delta t_e = \frac{1}{n}\sqrt{\sum_{i=1}^{n} \Delta t_{e_i}^2} \qquad (24)$$

Besides the metrics mentioned so far, we also consider the average size of the ciphertexts S_c, the parameters to be stored permanently for encryption $S_{p,e}$ or decryption $S_{p,d}$ and temporarily for group updates S_t. Here, $S_{p,e}$ consists of the required memory for PK, and $S_{p,d}$ for the required memory for S_i, ζ_1, and ζ_2. S_t consists of the required memory to store γ. Since we can accurately determine the required storage space, we assume the standard deviation of the measured sizes as an error.

C. Measurement Setup

The objective of this paper is to analyze the performance of the group-oriented encryption scheme for dynamic groups from [1] in an IoT context. Hence, we apply an evaluation in an IoT-typical scenario using microcontrollers as hardware for the group members. As a typical example of IoT hardware, we choose the ESP32 microcontroller, since it is wide-spread and used in various IoT systems, for example, automated solar water pumping systems [9], smart surveillance [10], or smart saline level monitoring [11]. The ESP32 is a 32-bit microcontroller from Espressif Systems and a so-called System-on-a-Chip. It has a 240 MHz dual-core CPU, 512 kB RAM, and the ability to establish a 2.4 GHz Wi-Fi connection. For ease of use, we used the developer board version of the ESP32 because it can be connected conveniently via USB, allowing for conveniently flashing and reading out the serial pin of the ESP32 using a computer. For simplicity, we see the serial pin as an external interface to the console output of the ESP32.

In the following, we explain the energy efficiency measurement setup used in the scenario of encrypting messages (workload pattern (1)) and then show how we modified it for the other scenarios.

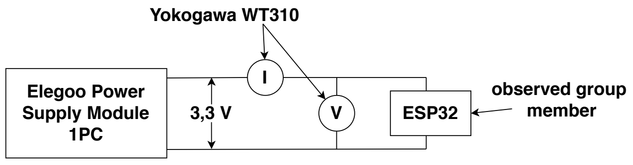


Fig. 2: Circuit diagram of the measurement setup

Figure 2 illustrates the setup for power measurements. In this setup, we use the Elegoo Power Supply Module 1PC to power the ESP32, since it can directly provide the required 3,3V for the ESP32. We use a Yokogawa WT310 power meter to measure the power consumption of the ESP32.

For implementing the actual schema, we require a library that allows the calculation of bilinear mappings. For this purpose, we decided to use the pbc library in version 0.5.14, because it is one of the few standard libraries for this purpose and is present in other applications like Boneh-Lynn-Shacham short signatures or Hess identity-based signatures [12]. Since the pbc library bases on the gmp library, we also use the gmp library in version 6.1.2 [13]. As a concrete pairing, we used the Type A pairings from the pbc library.

For measurement setups for workloads that do not require power measurements, the ESP32 was connected directly to the laptop via USB for convenient access to its serial port. Besides, we measured all metrics describing only the required storage space of specific parameters directly on the laptop performing the corresponding necessary group operations.

V. Practical Evaluation

In the course of this section, we first evaluate the measured memory requirements and then present the calculation times and energy efficiency in more detail.

A. Storage Analysis

We performed all storage measurements on a PC, as we are only interested in the particular memory requirements that are platform-independent. For this reason, the measurements have been made on the PC to allow more measurements.

A group member must permanently store the parameter $S_{p,e}$, consisting of Pk to encrypt messages and respectively $S_{p,d}$ to decrypt messages, consisting of ζ_1, ζ_2 and S_i. Thereby for storing $S_{p,e}$ exactly 2048 bits of memory are required and for $S_{p,d}$ a total of 6302 ± 2 bits (rounded up to whole bits). Furthermore, the required memory space for $S_{p,e}$ has the fluctuation of 2 bits due to the non-static memory requirement of S_{i_7}. The corresponding memory requirements were analyzed based on 1000 groups consisting of only one member since the observed parameters are independent of the actual group size.

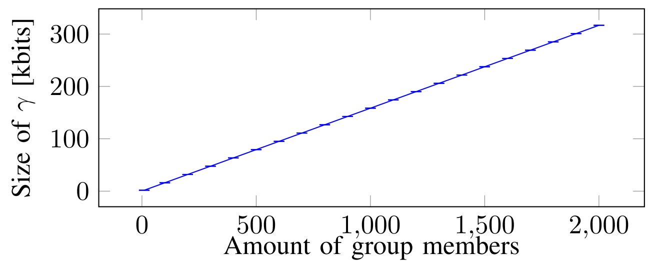


Fig. 3: The size of γ for different group sizes

A group member must temporarily store the information S_t consisting of γ to initially join a group or to perform group update operations. Thereby, the amount of memory required for storing γ depends on the size of the group, as illustrated in Figure 3. The size of γ for a concrete group size s_g was determined by analyzing the size of γ for 1000 groups of size s_g. Figure 3 shows practically that the theoretical assumption from Section III, that the size of γ increases linearly with group size, is correct in the considered measuring range. The practical evaluation also allows the statement that on average γ increases by 159 bits (rounded up) per group member in the considered measuring range.

As the last parameter regarding memory requirements, we analyzed the size of ciphertexts. Thereby we have encrypted messages consisting of up to 50 bytes and received within this range, independent of the concrete chosen message size, fixed ciphertext sites of precisely 3072 bits. The reason for the chosen measuring range is that the majority of messages in IoT are small and often less than 40 bytes [14]. At this point, it is noteworthy that constant chipper texts are not a matter of course in group encryption methods, and their length is often related to the number of group members, for example, in [6].

Overall, the evaluation of the considered memory requirements shows that the memory requirements, in the considered measuring range, are to be regarded as constant, apart from the linearly increasing memory requirement of γ.

B. Computation Time Analysis

In the following, we present the measured calculation times for the pure encryption t_e and decryption t_d and the execution of group update operations t_u. We performed the corresponding workloads (1), (2), and (3) on the ESP32.

We measured t_e and t_d for different message sizes, which is illustrated in Figure 4. Thereby the measuring range was selected as before. The measurement values illustrated in Figure 4 allow us to conclude that for the considered measurement range (i) it is acceptable to regard the times for encrypting and decrypting as constant, (ii) that decryption needs 1.389 ± 0.004 seconds and encryption needs 0.569 ± 0.058 seconds, and (iii) that decrypting messages takes more than twice as long as encrypting messages.

In addition to the times required for encryption and decryption, we measured the times required by the ESP32 to perform group update operations. These measured times are shown in

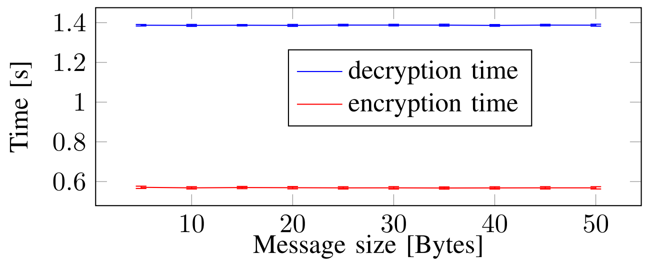


Fig. 4: Time needed for messages encryption/decryption of different sizes

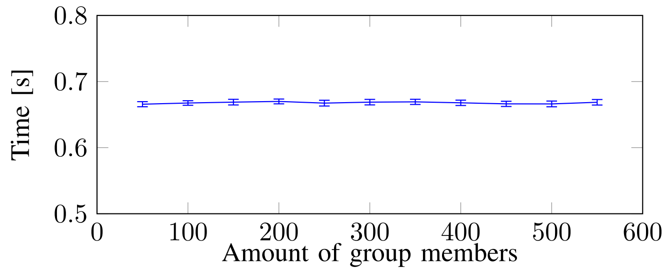


Fig. 5: Time needed to execute group updates for different group sizes

Figure 5 for group sizes between 50 and 550. The measurement range was limited because too large group sizes require larger contiguous memory areas than the ESP32 with the used libraries, and our implementation could not provide. Therefore, we decided to start with a group size of 50 measurements and to increase it in steps of 50 as long as the group update operation was successful. Figure 5 allows the conclusion that it is acceptable for the considered measuring range, to assume that the execution of group operations requires constant time independent of the group size. Thereby 0.669 ± 0.039 seconds are to be expected for the execution of an update operation.

Overall, our measurements confirm the assumption of constant calculation costs in the measuring range under consideration.

C. Energy Efficiency Analysis

In the following, we present the measured energy efficiency and power consumption of the scheme. We executed all measurements on the ESP32.

The general energy efficiency of the encryption and decryption procedures are determined using the workload patterns (1) and (2), and the corresponding energy efficiency values are illustrated in Figure 6. The measuring range was selected analogously to the previous subsection. Figure 6 allows us to derive two statements for the considered measuring range. First, decrypting messages is more energy consuming than encrypting messages. One might have already expected this statement from the different calculation times for decryption and encryption. However, different execution times alone would not have allowed this conclusion since, in general, a program that requires more time does not automatically require

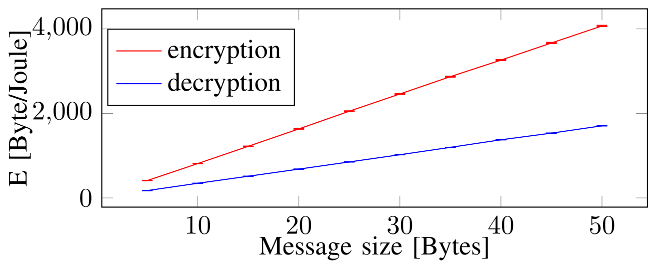


Fig. 6: Energy efficiency of decrypting/encrypting messages of different sizes

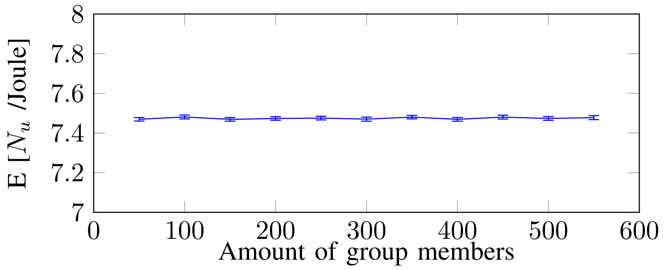


Fig. 7: Energy efficiency of group update operations for diffrent group sizes

more energy [15]. Second, the energy efficiency of encryption and decryption increases linearly with the message length. For this reason, from an energy point of view, it does not bring any benefits in practice to, e.g., divide a message into two messages and to encrypt and decrypt them separately.

In addition to decryption and encryption, we also measured the energy efficiency for group update operations separately on the ESP32 using workload (3). The measuring range was again selected analogously to the previous section, and the measured efficiency is depicted in Figure 7. This figure allows us to state that for the considered measuring range and within our measuring accuracy, it is acceptable to assume that the energy consumption for group update operations is constant and independent from the actual group size. Also, the figure allows us to make statements about the maximum number of possible group operations that would be possible in this measurement range for a battery-powered ESP32. For example, if the ESP32 would be powered by a CR123 A Lithium battery (which provides 4.65 kJ) [16], then (if we overestimate the group update operations per Joule to be 7.6), not more than 35340 update operations would be possible.

Finally, considering the energy usage of ESP32 in IDLE mode (workload pattern (4)) compared to continuous group updates (workload pattern (3)) with 550 users yields the overhead caused by keeping group information update to date. Since the throughput of IDLE is 0, we take the average energy consumption per second as a measure of comparison. The measurements resulted in 0.20504 $\pm$ 0.0028 Joule for IDLE and 0.21803 $\pm$ 0.00027 joules for group updates. Within the scope of the measurement accuracy, the overhead creates an extra energy consumption of less than 0.02 Joule.

VI. Related Work

In the domain of encryption procedures for efficient communication with more than one communication partner, many schemes emerged in recent years, e.g., [1], [5], [6], [17]–[19] to name some. The works differ significantly in their performance evaluation — some papers provide theoretical analysis [5], [6], [19], while others offer analysis based on empirical measurements with real hardware [1], [17] or simulated measurements [18].

Theoretically evaluated encryption schemes often originate from the crypto community that commonly use Landau notation in evaluation. While such a comparison helps estimating the complexity of developed algorithms, they are less useful for practitioners. Practical approaches to evaluation, however, often lack unified evaluation scenarios and metrics, which makes it difficult to compare them with one another. For instance, some evaluation scenarios target conventional computers such as desktop PCs or laptops (as in [1]), while others aim at IoT specific platforms and scenarios [17], [18]. We also chose an IoT setting because (i) IoT gradually becomes an integral part of our daily environment, (ii) IoT systems typically represent the target group of the considered encryption schemes by consisting of a large number of communication participants, and (iii) for the resource-limited IoT devices the performance of the methods is much more critical compared to scenarios with powerful computers. The considered performance aspects — such as the required storage space, calculation times, and energy efficiency — are also already used in the literature [17], [18]. However, our evaluation differs in (i) how we power the observed IoT device and (ii) how to determine the efficiency of energy usage. As a common practice in related work, IoT devices run on battery power, and the battery lifetime, battery status and the average current determine energy efficiency (e.g., [17], [18]). However, these approaches some disadvantages. For example, in the battery datasheets (e.g., see the datasheet of a CR123 A Lithium battery [16] used in [17]) the discharge curve typically shows a non-linear behavior, so only the current cannot be used as a measure of the consumed energy. Besides, the use of a battery thus leads to the fluctuating power supply, whose resistance also changes (e.g., for a changing internal battery resistance of a Lithium-Ion battery see [20]) and thus also influences the current. Although battery life is a metric that is of great practical interest, this metric requires additional information to interpret it. Thus, it is essential to know the complete history of the IoT device's behavior to interpret its battery lifetime.

To avoid the limitations of using battery lifetime and the average current for determining the energy efficiency, we decided to power our chosen IoT device with a constant power supply, as this allows us to determine the energy efficiency in a time stable measurement environment. This approach also allows us to make statements about individual operations and to indicate their actual energy consumption in Joules. That way, we can directly compare the energy consumption of two different operations, like how much energy the encryption of a message of 10 Bytes consumes in contrast to the encryption of a 100 Bytes message. In addition to the modified test setup, we have also specified the corresponding error formula for the metrics used to be able to make statements about the accuracy of the values determined.

VII. Conclusion

In this paper, we present the evaluation of the state-of-the-art group-oriented encryption scheme from [1]. Therefore, we describe a revised version of the approach, a measurement workflow, and a design of a reproducible hardware testbed. Using this evaluation environment, we analyze the performance of the encryption scheme in a typical IoT scenario. The results show that that encryption, decryption, and group updates need constant time and complete in less than two seconds. In terms of memory requirement for the permanently stored parameters, the storage requirement remains below 8.5 kbit. However, we identify a bottleneck of the scheme since the temporary parameter stored for group updates are increasing linearly with the group size.

For future work, we plan to extend our experiments in terms of network communication and, therefore, combine the scheme with IoT typical communication protocols like MQTT or OPC UA. Further, we already used real hardware in a controlled environment. For future work, it would be interesting to apply experiments in a real-world IoT system as this would integrate additional challenges, such as mobility of devices or fluctuation of the network connection. Additionally, in this paper, we analyzed the encryption scheme from Nishat et al. [1]. For future work, it would be interesting to compare other approaches, such as those mentioned in the related work section: [5], [6], [17]–[19]. Using such a knowledge base of experiments might help to derive general guidelines for the use of group-oriented encryption schemes for IoT systems. Those guidelines allow us to derive the best encryption scheme; moreover, those can be encoded in a self-aware computing approach [21] to automatically choose the best approach dynamically at runtime depending on characteristics of the group as well as the requirements for communication.

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) [22].

References

[1] K. Nishat *et al.*, "Group-oriented encryption for dynamic groups with constant rekeying cost," *Sec. and Commun. Netw.*, vol. 9, no. 17, Nov. 2016. [Online]. Available: https://doi.org/10.1002/sec.1593

[3] T. Stack, "Internet of Things (IoT) Data Continues to Explode Exponentially. Who Is Using That Data and How?" 2017, online available under https://blogs.cisco.com/datacenter/internet-of-things-iot-data-continues-to-explode-exponentially-who-is-using-that-data-and-how, Accessed on 24.01.2020.

[2] M. Hung, "Leading the IoT - Gartner Insights on How to Lead in a Connected World," 2017, online available under https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf, Accessed on 24.01.2020.

[4] Forbes, "Can a Fitness Tracker Save Your Life," 2019, online available under https://www.forbes.com/sites/insights-teradata/2020/04/21/how-disney-plus-personalizes-your-viewing-experience/#543938f83b6e, Accessed on 11.06.2020.

[5] B. Malek *et al.*, "Adaptively secure broadcast encryption with short ciphertexts." *IACR Cryptology ePrint Archive*, vol. 2010, p. 277, 01 2010.

[6] D. Boneh *et al.*, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Advances in Cryptology – CRYPTO 2005*, V. Shoup, Ed. Springer Berlin Heidelberg, 2005.

[7] Y. T. . M. Corporation, "WT300 Serie Digitale Leistungsmessgeräte," online available under https://tmi.yokogawa.com/de/solutions/products/power-analyzers/digital-power-meter-wt300/#Details, Accessed on 28.03.2020.

[8] S.P.E.C, "Power and performance benchmark methodology v2.2," 2014.

[9] S. Bipasha Biswas *et al.*, "Solar water pumping system control using a low cost esp32 microcontroller," in *2018 IEEE Canadian Conference on Electrical Computer Engineering (CCECE)*, 2018, pp. 1–5.

[10] P. Rai *et al.*, "Esp32 based smart surveillance system," in *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2019, pp. 1–3.

[11] D. Ghosh *et al.*, "Smart saline level monitoring system using esp32 and mqtt-s," in *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 2018, pp. 1–5.

[12] B. Lynn, "The Pairing-Based Cryptography Library," online available under https://crypto.stanford.edu/pbc/, Accessed on 24.06.2020.

[13] F. S. Foundation, "The GNU Multiple Precision Arithmetic Library," online available under https://gmplib.org/, Accessed on 24.06.2020.

[14] I. Management Association, *The Internet of Things: Breakthroughs in Research and Practice: Breakthroughs in Research and Practice*, ser. Critical explorations. IGI Global, 2017, page 509. [Online]. Available: https://books.google.de/books?id=7RshDgAAQBAJ

[15] E. Capra *et al.*, "Is software "green"? application development environments and energy efficiency in open source applications," *Information and Software Technology*, vol. 54, no. 1, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584911001777

[16] *CR 123 A Lithium Manganese Dioxide*, VARTA Microbattery GmbH, 9 2014, online available under http://https://products.varta-microbattery.com/applications/mb_data/documents/data_sheets/DS6205.pdf, Accessed on 24.06.2020.

[17] S. Kumar *et al.*, "JEDI: Many-to-many end-to-end encryption and key delegation for iot," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/kumar-sam

[18] M. Elhoseny *et al.*, "An energy efficient encryption method for secure dynamic wsn," *Security and Communication Networks*, vol. 9, no. 13, pp. 2024–2031, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1459

[19] R. Gay *et al.*, "Tight adaptively secure broadcast encryption with short ciphertexts and keys," in *Security and Cryptography for Networks*. Cham: Springer International Publishing, 2018, pp. 123–139.

[20] X. Wei *et al.*, "Internal resistance identification in vehicle power lithium-ion battery and application in lifetime evaluation," in *2009 International Conference on Measuring Technology and Mechatronics Automation*, vol. 3, 2009.

[21] Samuel Kounev et al., "The Notion of Self-aware Computing," in *Self-Aware Computing Systems*. Springer, 2017.

[22] T. Prantl *et al.*, "SIMPL: Secure IoT Management Platform," in *ITSec*, ser. 1st ITG Workshop on IT Security, 2020.