Towards a Group Encryption Scheme Benchmark: A View on Centralized Schemes with Focus on IoT

Thomas Prantl, Peter Ten, Lukas Iffländer, Stefan Herrnleben, Alexandra Dmitrenko, Samuel Kounev
{firstname.lastname}@uni-wuerzburg.de
University of Würzburg, Germany

Christian Krupitzer
christian.krupitzer@uni-hohenheim.de
University of Hohenheim, Germany

ABSTRACT

The number of devices connected to the Internet of Things (IoT) is continuously increasing to several billion nowadays. As those devices often share sensitive data, encryption of those data is an important issue. The pure volume of data and the complexity of communication patterns increases, and, accordingly, the importance of group encryption is recently gaining more importance. Still, the choice of the best-suited group encryption scheme for a specific application is complicated. Benchmarks can support this choice. However, while literature distinguishes three categories for the schemes (central, decentral, and hybrid), a one-fits-all benchmark seems challenging to achieve. In this paper, we go the first step towards a structured benchmark for group encryption schemes by presenting a benchmark for centralized group encryption schemes in an IoT scenario. To this end, our benchmark includes a description of workloads, a baseline scheme, a measurement setup, and metrics while also considering the requirements and features of centralized group encryption schemes.

KEYWORDS

Group Encyrption Scheme, Benchmark, Energy Efficiency

ACM Reference Format:
Thomas Prantl, Peter Ten, Lukas Iffländer, Stefan Herrnleben, Alexandra Dmitrenko, Samuel Kounev and Christian Krupitzer. 2021. Towards a Group Encryption Scheme Benchmark: A View on Centralized Schemes with Focus on IoT. In *Proceedings of the 2021 ACM/SPEC International Conference on Performance Engineering (ICPE '21), April 19–23, 2021, Virtual Event, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3427921.3450252

1 INTRODUCTION

The number of deployed devices in the Internet of Things (IoT) is continuously increasing, and there will be 20 billion IoT devices this year [15]. Thus, the pure volume of data and the complexity of communication patterns, typically involving groups of devices interested in sharing data with constantly changing group membership dynamics, increase. Besides, feature requirements for the encryption scheme depend on the use case. An IoT-supported medical study, for example, may require that when adding group members in charge of data analytics in the course of data collection, they can also decrypt previously encrypted group messages. IoT-supported platooning (e.g., [20]), a group of vehicles that can travel closely together, safely at high speed, may, on the other hand, require that new cars joining a platoon cannot decrypt messages from former platoon members for privacy reasons.

Accordingly, with the increasing number of IoT devices, the importance of group encryption schemes and their features increases. Especially in domains with resource-constrained devices such as IoT, a scheme's performance and requirements are essential issues. However, the performance evaluations of group encryption schemes mostly use the Big O notation, not considering all involved actors (e.g., [6, 13]). As discussed in [24], existing studies with real measurements lack suitable hardware testbeds and well-defined metrics. This shortcoming limits their analyses' expressiveness missing key performance metrics for practitioners (e.g., calculation times or storage requirements) and not reporting precise measured values but partly estimating them using Big O notation. Such analysis results combined with an overview of provided features and requirements, if available, would peculiarly support developers in the IoT domain to judge the suitability of group encryption schemes. Ideally, choosing the best group encryption scheme for a specific application is founded on benchmark results. Such a benchmark must span the whole bandwidth of group encryption schemes. The literature distinguishes three scheme categories: centralized, decentralized, and hybrid schemes [7]. Designing a single benchmark that simultaneously covers all different categories is challenging due to their fundamental differences.

In this paper, we take the first step towards benchmarking group encryption schemes by presenting a benchmark for centralized group encryption schemes with focusing on IoT. Our contributions include:

- the description of workloads, measurement setup, metrics, requirements, and features for a centralized group encryption scheme benchmark;
- the modification of Boneh's et al. encryption scheme supporting dynamic groups and flexible load distribution;
- the benchmarking and comparison of (i) two implementations of Boneh's et al. modified scheme, (ii) an improved version of the group encryption scheme by Nishat et al. [23] from [24], and (iii) our SKDC baseline.

In the remainder of this paper, Section 2 describes the modifications to the scheme presented by Boneh et al. [6] to distribute the load among the involved actors flexibly. Section 3 presents our benchmark for centralized group encryption schemes. In Sections 4 and 5,

ICPE '21, April 19–23, 2021, Virtual Event, France

ACM ISBN 978-1-4503-8194-9/21/04…$15.00
https://doi.org/10.1145/3427921.3450252

we benchmark and evaluate the schemes concerning their features, requirements, and performance in our testbed. In Sections 4 and 5, we (i) analyze the two implementations of the modified scheme proposed by Boneh et al. and the corrected scheme [24] proposed by Nishat et al. [23] in our testbed concerning their features, requirements, and performance, and (ii) compare them with the baseline scheme.

2 BONEH+: A MODIFICATION OF THE BONEH ET AL. SCHEME

This section introduces two different implementations of Boneh's et al. scheme [6] modified to support dynamic groups and flexible load distribution. Therefore, we present (i) the involved actors, (ii) the encryption and decryption of messages, (iii) the initial creation of a group consisting of n members, and (iv) changing the group composition. We present our modifications to (i) show different implementations of the same scheme, (ii) present the impact to requirements and workloads for environment and actors, and (iii) motivate our benchmark design. In the following, the designation Boneh+ denotes the modified variants. Further, we differentiate the modified variants in a fat client (indexed with "Fat Client") or a thin client (indexed with "Thin Client").

Involved Actors: The actors consist of the group members, who should communicate securely with each other, and a central instance (CI), which controls group creation and management. In general, each group member could take over this role. However, we assume a dedicated CI in the following. The CI defines the parameters for decryption and encryption, generates the keys, and distributes them to the group members for managing group memberships. Since the CI creates and thus knows the group key, the CI can read all group messages and, therefore, must be a trustworthy party. Secure communication must be available initially between the CI and each group member for the inclusion into a group. Thereby, using a secure channel ensures confidentially of the transmitted information. A straightforward approach is directly connecting devices. For subsequent communication between a group member and the CI, an insecure connection suffices as long as it guarantees reliable delivery. Otherwise, members might, for example, miss member removal updates and continue to encrypt messages in a fashion decryptable by excluded members. For *Boneh+* and the rest of this paper, we assume that messages are neither delayed, replayed, falsified, nor intercepted.

Initial Group Creation: The initial group creation with n group members requires the CI to determine $\Gamma = \{\zeta, I, g, \alpha, \gamma, \}$. Thereby, ζ comprises information about the used encryption scheme in conjunction with the group key to encrypt and decrypt the actual group messages. The parameter I consists of selecting a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ used for encrypting and decrypting the group key. The CI randomly chooses the parameters α and γ from $\mathbb{Z}_p^*$ and a generator g of G_1. Using Γ and the Equations 1 and 2, the CI calculates the public key PK and all secret user keys SK_i.

$$PK = (\underbrace{g}_{g_0}, \underbrace{g^{\alpha^1}}_{g_1}, \underbrace{g^{\alpha^2}}_{g_2}, \ldots, \underbrace{g^{\alpha^n}}_{g_n}, \ldots, \underbrace{g^{\alpha^{2n}}}_{g_{2n}}, \underbrace{g^{\gamma}}_{v}) \quad (1)$$

$$SK = (\underbrace{g_1^{\gamma}}_{SK_1}, \underbrace{g_2^{\gamma}}_{SK_2}, \ldots, \underbrace{g_n^{\gamma}}_{SK_n}) \quad (2)$$

The CI distributes the secret keys SK_i to the respective group members with I and ζ via a secure channel.

Each group member must know its values of $u_{c,i}$, defined in Equation 3, and g_i to decrypt later the encrypted group key. For the calculation of $u_{c,i}$, each group member needs to store the value of the variable a_i, defined in Equation 4, in addition to its secret key SK_i. There are two possibilities for how individual group members can receive this value. First, the CI can broadcast the public key PK to all members, and each member derives its value of $u_{c,i}$. We call the variant *Boneh+*$_{\text{Fat Client}}$. Alternatively, the CI computes $u_{c,i}$ individually for each group member and unicasts it to the respective group member. Note that the second variant still requires the CI to communicate the respective value of g_i to each member. We designate the second variant *Boneh+*$_{\text{Thin Client}}$. Neither option requires a secure channel for the transmissions.

$$u_{c,i} = SK_i * a_i \quad (3) \qquad a_i = \prod_{m=1}^{n} g_{n+1-m+i}/g_{n+1} \quad (4)$$

Since each group member now has all the information to decrypt the encrypted group key C, the CI can calculate it using Equation 5 and distribute it. Thereby, the distribution does not require a secure channel. The calculation of C using Equation 5 requires the CI to calculate the value of u_s using Equation 6 and choose t randomly from $\mathbb{Z}_p^*$.

$$C = (C_0, C_1) = (g^t, u_s^t) \quad (5) \qquad u_s = v * \prod_{i=1}^{n} g_{n+1-i} \quad (6)$$

The individual group members can now calculate the group key K using Equation 7 and the variables g_i, C, and $u_{c,i}$.

$$K = e(g_i, C_1)/e(u_{c,i}, C_0) \quad (7)$$

Revocation of Group Members: The i-th group member's revocation process requires updating the list of revoked group members R and establishing a new group key among the remaining group members. Therefore, the CI must determine a new encrypted group key C' using Equation 8, wherefore CI chooses $t' \in \mathbb{Z}_p^*$ randomly and calculates u_s' using Equation 9. Thereby, g_r in Equation 9 stands for the g_i value of the revoked group member. After the calculation of C', the CI distributes the new encrypted group key C'.

$$C' = (C_0^{'}, C_1^{'}) = (g^{t'}, u_{s'}^{t'}) \quad (8) \qquad u_s' = u_s/g_r \quad (9)$$

The remaining group members can calculate the new group key using Equation 16, using u_s' in addition to C'. Thereby, u_s' originates from Equation 10, for which each group member needs its $g_{n+1+i-r}$. For the transmission of the value of $g_{n+1+i-r}$, there are again two approaches. (i) The *Boneh+*$_{\text{Fat Client}}$ consists of broadcasting PK' and the number r of the revoked member. (ii) In *Boneh+*$_{\text{Thin Client}}$, the CI sends the value of $g_{n+1+i-r}$ from PK' individually to each group member. Again, both approaches do not require a secure channel.

$$u_{c,i}' = u_{c,i}'/g_{n+1+i-r} \quad (10)$$

Addition of Group Members: The addition process of a new group member requires the CI to first update PK to PK' according to Equation 11. Next, the CI must calculate a secret key SK_{n+1} for the

new group member using Equation 12 and transmit SK_{n+1}, I, and ζ to it via a secure channel.

$$PK' = (\underbrace{g}_{g_0}, \underbrace{g^{\alpha^1}}_{g_1}, \ldots, \underbrace{g^{\alpha^{2n}}}_{g_{2n}}, \underbrace{g^{\alpha^{2n+1}}}_{g_{2n+1}}, \underbrace{g^{\alpha^{2n+2}}}_{g_{2n+2}}, \underbrace{g^{\gamma}}_{v}) \quad (11)$$

$$SK' = (\underbrace{g_1^{\gamma}}_{SK_1}, \underbrace{g_2^{\gamma}}_{SK_2}, \ldots, \underbrace{g_n^{\gamma}}_{SK_n}, \underbrace{g_{n+1}^{\gamma}}_{SK_{n+1}},) \quad (12)$$

For the meanwhile enlarged group, the CI must determine a new encrypted group key C' using Equation 13. This task requires the CI to select t' randomly from $\mathbb{Z}_p^*$ and calculate u_s' using Equation 14.

$$C' = (C_0', C_1') = (g^{t'}, u_s^{t'}) \quad (13) \qquad u_s' = u_s * g_{n+1} \quad (14)$$

In order for the group members to decrypt the new encrypted group key, the existing group members must update their decryption information, and the new group member must first receive it. The existing group members can do this using Equation 18 based on s_i from Equation 15. Again the *Boneh+*$_{\text{Fat Client}}$ and *Boneh+*$_{\text{Thin Client}}$ vary regarding the transmission of this value. In the *Boneh+*$_{\text{Fat Client}}$ variant, the CI broadcasts PK' and R to all members, who use them to calculate the value of s_i themselves. Thereby R is the list of removed group members. The *Boneh+*$_{\text{Thin Client}}$ variant consists of the CI calculating the value for all group members and unicasting it to the corresponding group member.

The new group member also needs the decryption information for the encrypted group key, which is, in this case, the value of $u_{c,n+1}$. Also, there are two different ways for the new member to get this value. According to Equation 19, the CI calculates $u_{c,n+1}$ and unicasts it to the new group member in the *Boneh+*$_{\text{Thin Client}}$ variant. In the *Boneh+*$_{\text{Fat Client}}$, the CI also broadcasts PK' to the new group member so to calculate $u_{c,n+1}$ using Equation 3.

$$s_i = g_{n+1} * g_{n+1+i}/\delta_i \quad (15) \qquad K' = e(g_i, C_1')/e(u_{c,i}', C_0') \quad (16)$$

$$\delta_i = \begin{cases} 1, & \text{the i-1.th group member was removed} \\ g_{n+2}, & \text{the i-1.th group member was not removed} \end{cases} \quad (17)$$

$$u_{c,i}' = u_{c,i} * s_i \quad (18) \qquad u_{c,n+1} = u_{c,n}' * g_{n+2+i}/g_{n+1} \quad (19)$$

With the updated information, the old and new group members can decrypt the new encrypted group key K' using Equation 16.

3 TOWARDS A GROUP ENCRYPTION SCHEME BENCHMARK FOR IOT

To come closer to a cross-category benchmark for group encryption schemes, we propose in this section a benchmark for centralized schemes with a focus on IoT, which we plan to extend towards decentralized and hybrid schemes in the future. Therefore, we (i) highlight the importance of taking centralized schemes' performance, their requirements, and the features they provide into account. We subsequently present (ii) workload patterns, (iii) an IoT typical measurement setup, (iv) performance metrics, and (v) a baseline scheme.

3.1 Requirements and Features

Choosing a group encryption scheme is not just about pure performance aspects such as the time it takes to join a group but also whether the respective scheme can fulfill all requirements and required features of an underlying use case. Therefore, a cross-category benchmark must include (i) the requirements (like network topology) imposed by the encryption scheme and (ii) the features (like forward secrecy) it provides in addition to the performance analysis. We propose initial requirements and features, which a benchmark for centralized group encryption schemes should consider.

Requirements: To determine the requirements of group encryption schemes, we initially analyzed the two implementations of the modified variant of the scheme proposed by Boneh et al., presented in Section 2. By doing so, we were able to identify that the group operations each have different requirements regarding topology and confidentiality of communication. It is essential for the respective group operation, whether the topology allows uni- and broadcasts. This property is crucial for IoT scenarios since they typically rely on broadcast communication protocols where the receiver and sender do not even need to know each other. Regarding communication confidentiality, some group operations must exchange their information through confidential channels, while other operations can also use public channels.

Features: We base the determination of group encryption scheme features on the analysis of [4, 24]. We determined three features: group size limit, group backward, and forward secrecy. We briefly explain the respective features in the following. The group size limit describes whether group update operations for a group of any size are limited. In [24], we have shown that this limit frequently applies when group members have to temporarily store a parameter that increases linearly with the group size to perform group operations. Therefore, this temporary parameter limits the size of realizable groups for the group members. Supposing a scheme satisfies the backward secrecy or forward secrecy properties, this means that a new group member does not have access to data transmitted before joining or after leaving the group [9]. We assume that this backward/forward secrecy is present when the addition/revocation of group members triggers an immediate corresponding adjustment. Not all schemes fulfill this property. For example, the work in [17] does not support backward secrecy.

3.2 Workload Pattern

The examples *Boneh+*$_{\text{Thin Client}}$ and *Boneh+*$_{\text{Fat Client}}$ from Section 2 show that multiple ways exist to distribute group management duties. For this reason, in our workload pattern, we differentiate by the actor, which performs a group management operation to capture the entire workload of the system actor precisely. In terms of group management, we assume the following three operations: group creation, member addition, and revocation. Regarding the example of group creation in Section 2, the respective group management operations can occur in different phases that can also build upon each other. Since the best practice is to deploy IoT devices into their operation area with predefined confidential information that allows joining a group, we differentiate the group creation between a so-called preparatory and operational phase. We also consider

them in our workload pattern since these phases separate the execution of specific workloads. Based on these considerations, we now define the group management workload pattern WP_{man} in Definition 3.1. Since group management aims at enabling group members to exchange encrypted messages, we also define a workload pattern in Definition 3.2, considering the cryptographic operations of decryption and encryption.

Definition 3.1. $\mathrm{WP}_{manage}(A, P, O, N)$: Actor A performs N times the necessary calculations of the group operation O in phase P.

Definition 3.2. $\mathrm{WP}_{crypt}(C, A, B, N)$: Actor A performs N times the cryptographic operation C on a message, which initially consists of B Bytes.

3.3 Measurement Setup

A realistic example scenario and the corresponding measurement setup are necessary to measure centralized group encryption methods on real hardware. Since IoT scenarios typically consist of many devices that communicate in a group, we have chosen such an example scenario. This choice allows us to reuse the measurement setup from [24] extended by the CI component. Our new measurement setup for reproducible measurements (see Figure 1) consists of the following four components: an observed group member, its power supply, a power meter to measure its energy consumption, and the CI.

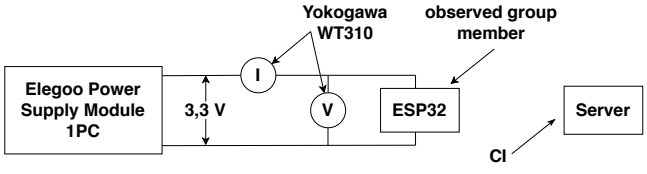

Figure 1: Circuit diagram of the measurement setup

For realizing the measurement setup, we propose using the ESP32 32-bit microcontroller for the group member. It is widespread and a part of various IoT systems, for example, automated solar water pumping systems [5]. When choosing a power meter, its accuracy must match the typical small voltages and current IoT devices' currents. Therefore, we use the Yokogawa WT310 as a power meter. According to the manufacturer, its accuracy in the corresponding measuring range is ±(0.1% of reading + 0.0006 Watt) [10], suitable for IoT devices. As CI, we decided to use a laptop because we wanted to analyze a standard laptop's suitability in the CI role for centralized group encryption procedures. Specifically, we are using a Lenovo B50-50 80S2004AGE with an Intel® Core™ i3-5005U 2x 2.00 GHz, Intel® HD Graphics 5500 Shared Memory, 4 GB RAM, and 500 GB HDD running Ubuntu 16.04.4 LTS.

3.4 Metrics

We generalize the metrics from [24] by introducing the CI component. Thereby, our metrics focus on three aspects: storage requirements, computation times, and energy efficiency. Based on these categories, we present the following metrics.

Storage Requirements: We consider the average memory consumption m for parameters permanently or temporarily stored by the group members or the CI to perform certain group operations and store ciphertexts. We add the corresponding indices to m to distinguish which case we are dealing with as follows: For ciphertexts, we add the index c. In all other cases, we add two indices. The first index distinguishes between storage on a group member or the central instance (gm or ci). The second indicates whether the storage requirement is temporary or permanent (tmp or per). For example, we denote the permanent storage space required for a group member with $m_{gm,per}$. We use the standard deviation of the measured size as the error for storage requirements since we can accurately determine the required storage space.

Computation Times: In Equation 20, we define the computation time $\bar{t}_A$ for an action A as the average time required to execute A. Assuming that A' is the encryption of a message by a group member, $\bar{t}_{A'}$ comprises encrypting a message n times and is the average of the corresponding times $t_{A',1}, \ldots, t_{A',n}$, needed for the single encryptions. We calculate its error using Gaussian error propagation according to Equation 21 to determine the accuracy of the calculation time. Here, $\Delta t_{A,i}$ stands for the accuracy of the time required to perform the i-th action A, which would be the accuracy of the time required for the i-th encryption. Since we assume that we can determine action A's duration with the same accuracy Δt_A, this simplifiest Equation 21 to Equation 22.

$$\bar{t}_A = \frac{1}{n}\sum_{i=1}^{n} t_{A,i} \quad (20) \qquad \Delta\bar{t}_A = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\Delta t_{A,i}^2} \quad (21)$$

$$\Delta\bar{t}_A = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\Delta t_{A,i}^2} = \frac{1}{n}\sqrt{n * \Delta t_A^2} = \frac{\sqrt{n} * \Delta t_A}{n} \quad (22)$$

Energy Efficiency: In line with SPEC specifications [26], we define the energy efficiency E in Equation 23 as the throughput T_A (see Equation 28) to power consumption W ratio. We calculate the accuracy of energy efficiency using Gaussian error propagation (see Equation 24).

$$E = \frac{\text{Throughput}}{\text{Power Consumption}} = \frac{T_A}{W} \quad (23)$$

$$\Delta E = \sqrt{\frac{\Delta T_A^2}{W^2} + \frac{T_A^2 * \Delta W^2}{W^4}} \quad (24)$$

We require the average power consumption per second W from Equation 25 to calculate the energy efficiency. In this equation, n stands for the measurement duration in seconds and W_i for the power consumption during the ith second. We determine W's accuracy in Equitation 26 using Gaussian error propagation to indicate an error range for the actual value of W. Thereby, ΔW_i is the accuracy of the measured power consumption during the i-th second.

$$W = \frac{1}{n}\sum_{i=1}^{n} W_i \quad (25) \qquad \Delta W = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\Delta W_i^2} \quad (26)$$

For our proposed measurement setup, the power measurement error of the used Yokogawa WT310 is ±(0.1% of reading + 0.2% of range) according to the manufacturer [10]. Thereby the range error in our case is 0.0006 Watt because we set the measuring ranges

	SKDC [21]				*Nishat* [23, 24]				*Boneh+*$_{\text{Fat Client}}$				*Boneh+*$_{\text{Thin Client}}$			
	Topology		Confidential		Topology		Confidential		Topology		Confidential		Topology		Confidential	
	unicast	broadcast	yes	no	unicast	broadcast	yes	no	unicast	broadcast	yes	no	unicast	broadcast	yes	no
Deployment	✗		✗		✗		✗		✗		✗		✗		✗	
Operational	✗			✗		✗		✗		✗		✗	✗			✗

Table 1: Requirements of the group creation operation of the centralized group encryption schemes *SKDC*, *Nishat*, *Boneh+*$_{\text{Fat Client}}$ and *Boneh+*$_{\text{Thin Client}}$.

to 3V and 100mA. Thus, for our measurement setup, ΔW can be calculated using Equation 27.

$$\Delta W = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(0.1\% * W_i + 0.0006 * W)^2} \tag{27}$$

The calculation of energy efficiency still requires the determination of throughput T_A. We define the throughput in Equation 28 as the weighted number of performed actions A during a period t_p. We determine the throughput for only one specific action at a time. In Equation 28, $|A|$ stands for the number of actions A performed and W_A for the weighting factor of action A. We define W_A as follows: W_A is the number of decrypted or encrypted bits for decryption and encryption actions. For all other actions, we set W_A to the value 1. Using Gaussian error propagation, the error of T_A results from equation 29, where Δt_p stands for t_p's accuracy, and we assume that all actions were successful.

$$T_A = \frac{W_A * |A|}{t_p} \tag{28}$$

$$\Delta T_A = \frac{W_A * |A| * \Delta t_p}{t_p^2} \tag{29}$$

3.5 Baseline Group Encryption Scheme

Over the years, many different schemes suitable for comparison in our benchmark emerged. However, most overlook straightforward group encryption implementations, such as the simplest way according to literature: *Simple Key Distribution Center* (*SKDC*) [6]. Therefore, in the following, we propose and present a realization of *SKDC* as a comparison baseline for all group encryption schemes. All other schemes, which are often more complex (e.g., in terms of implementation effort or mathematical foundations) than *SKDC*, must compensate for their extra effort with better performance, additional features, or lower environmental requirements.

In short, *SKDC* works as follows. The involved actors are the CI and the group members. Each group member has a unique symmetric key known to the CI. These symmetric keys enable encrypted communication between the CI and the respective group member. The CI determines a symmetric group key, which the group members use for their group communication, to enable encrypted communication between the group members. The CI securely distributes the group key to the individual group members by encrypting the group key with the respective group member's symmetric key. Adding or removing members is equivalent to creating a new corresponding group.

4 FEATURE- AND REQUIREMENT-DRIVEN ANALYSIS OF THE SCHEMES

Group encryption methods can encrypt messages, but additional properties are relevant, like forward and backward secrecy. Each of the group encryption methods imposes different requirements on the environment and provides various features. Accordingly, selecting a group encryption method should consider those properties besides pure performance values. This section determines the features and requirements of the schemes *SKDC*, *Nishat*, *Boneh+*$_{\text{Fat Client}}$, and *Boneh+*$_{\text{Thin Client}}$. Due to space constraints, we focus our analysis on the group creation process. This kind of analysis can support developers in choosing the best-fitting group encryption method for their application.

Table 1 summarizes each scheme's requirements and shows that (i) *SKDC* and *Boneh+*$_{\text{Thin Client}}$, and (ii) *Boneh+*$_{\text{Fat Client}}$ and *Nishat* have the same requirements. In particular, *Boneh+*$_{\text{Thin Client}}$ and *SKDC* cannot use broadcast communication since they send individualized messages for each group member. On the other hand, *Nishat* and *Boneh+*$_{\text{Fat Client}}$ can broadcast the messages to the group members because all members get the same messages in almost all group operations, except for the deployment phase. Furthermore, all schemes require a confidential channel for the deployment phase only.

Table 2 summarizes the respective schemes' features. All schemes provide backward and forward secrecy since they require immediate action to update encryption when adding or removing a group member. Thus, a new member cannot decrypt previously encrypted messages, and a removed member cannot decrypt messages encrypted after its removal. In terms of features, the considered schemes differ only by the group size limitation. Hence, for the schemes *SKDC* and *Boneh+*$_{\text{Thin Client}}$, the group size is unlimited

Schemes / Features	*SKDC* [21]	*Nishat* [23, 24]	*Boneh* Thin Client	*Boneh* Fat Client
group size limit	unlimited	limited	unlimited	limited
backward secrecy	yes	yes	yes	yes
forward secrecy	yes	yes	yes	yes

Table 2: Features of the centralized group encryption schemes *SKDC*, *Nishat*, *Boneh+*$_{\text{Fat Client}}$ and *Boneh+*$_{\text{Thin Client}}$

regarding adding and removing members at any time. This assumption does not apply to the schemes *Nishat* and *Boneh+*$_{\text{Fat Client}}$.

5 PERFORMANCE ANALYSIS OF THE SCHEMES

Applying our benchmark's methodology and metrics, we now determine the performance for the schemes *Boneh+*$_{\text{Fat Client}}$, *SKDC*, *Nishat*, and *Boneh+*$_{\text{Thin Client}}$ based on corresponding measurements in our measurement setup. Due to space constraints, we restrict our analysis of the group management operations to the group creation process. For this purpose, we (i) provide all necessary implementation information, (ii) analyze the group creation operation, and (iii) summarize our observations made during the evaluation.

5.1 Implementation Details

We require a library to calculate bilinear mappings for implementing the schemes *Boneh+*$_{\text{Fat Client}}$, *Boneh+*$_{\text{Thin Client}}$, and *Nishat*. 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 [22]. Since the pbc library depends on the gmp library, we also use the gmp library in version 6.1.2 [11]. Specifically, we use the Type A pairings from the pbc library.

The schemes *SKDC*, *Boneh+*$_{\text{Fat Client}}$, and *Boneh+*$_{\text{Thin Client}}$ require a symmetric encryption scheme, for which we chose AES-CBC-256 because of its approval by the U.S. government and standardization [14]. For the implementation of AES-CBC-256, we used OpenSSL [12] on the Linux laptop. On the ESP32, the firmware already provides the corresponding function.

5.2 Group Creation Performance Analysis

Regarding group creation, we analyze the calculation times, energy efficiency, and storage requirements. Thereby, we start with the CI analysis, followed by the analysis of the group members.

5.2.1 Computational Time. We start our analysis with the CI's calculation times to create a group in the deployment and operational phase. Figures 2 and 3 illustrate the calculation times. The results indicate that the calculation time for all presented methods increases linearly with the group size during the deployment phase. Additionally, we can say that *SKDC* performs best in this phase, followed by *Boneh+*$_{\text{Fat Client}}$ and *Boneh+*$_{\text{Thin Client}}$, which are identical for this phase (and are therefore combined in Figure 2 as *Boneh+*). However, this ranking flips in the operational phase. In this phase, *Nishat* performs best, followed by *Boneh+*$_{\text{Fat Client}}$, then *Boneh+*$_{\text{Fat Thin}}$, and finally *SKDC*.

For the group member analysis, we consider only the operational phase because the deployment phase consists solely of storing data. Figure 4 shows the time required by group members to join a group in the operational phase. For the selected measuring range *SKDC*, *Nishat* and *Boneh+*$_{\text{Thin Client}}$ require constant calculation times when creating groups, whereas *Boneh+*$_{\text{Fat Client}}$'s calculation times increase with the group size. Again, a ranking of the schemes shows that *SKDC* scores best, followed by *Nishat*, *Boneh+*$_{\text{Thin Client}}$, and *Boneh+*$_{\text{Fat Client}}$.

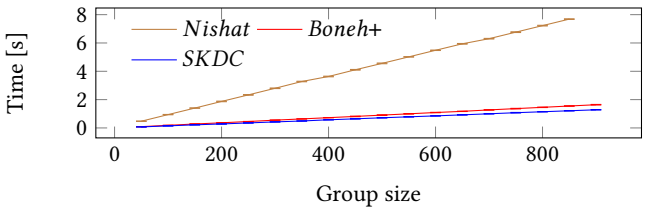

Figure 2: Computation times by CI during the group creation operation in the deployment phase.

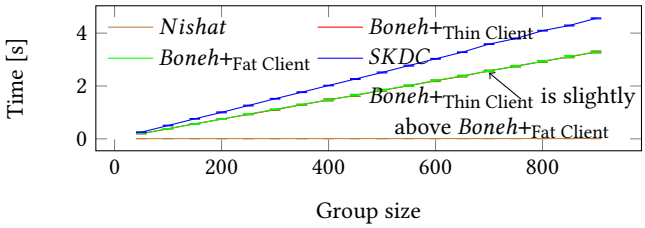

Figure 3: Computation times by CI during the group creation operation in the operational phase.

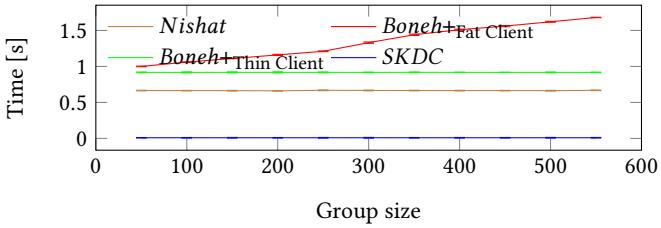

Figure 4: Computation times of group members during the group creation operation in the operational phase

5.2.2 Energy Efficiency. Due to space limitations, we only analyze the group members' energy efficiency for the group creation operation. Figure 5 illustrates that for *Nishat*, *SKDC*, and *Boneh+*$_{\text{Thin Client}}$, the energy efficiency is independent of the group size, whereas, for *Boneh+*$_{\text{Fat Client}}$, the energy efficiency decreases when the group size increases. For the measuring range, the individual schemes rank from worst to best for energy efficiency: *Boneh+*$_{\text{Fat Client}}$, *Boneh+*$_{\text{Thin Client}}$, *Nishat*, and *SKDC*. Thus, the baseline procedure again performs better than the other schemes in terms of energy efficiency on the group creation operation's member-side.

5.2.3 Memory Consumption. Finally, we analyze the memory requirements, where we limit our analysis on the CI due to space constraints. Figures 6 and 7 illustrate the measurements regarding temporary and permanent memory requirements. These figures show that the temporary memory requirement of *Boneh+*$_{\text{Fat Client}}$,

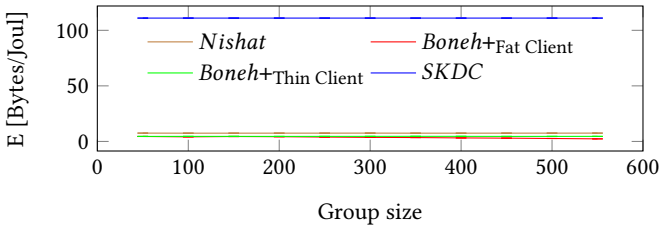

Figure 5: Energy efficiency E of group creation on group member side

Boneh+$_{\text{Thin Client}}$, and *SKDC* is independent of the group size. In contrast, for *Nishat*, the memory increases linearly with the group size (see Figure 6). Regarding the temporary memory requirements, the schemes rank from worst to best as follows: *Boneh+*$_{\text{Fat Client}}$, *Boneh+*$_{\text{Thin Client}}$, *SKDC*, and *Nishat*. For the permanent memory requirement, however, for all procedures the memory requirement increases linearly with the group size, resulting in the following ascending ranking: *Nishat*, *Boneh+*$_{\text{Fat Client}}$, *Boneh+*$_{\text{Thin Client}}$, and *SKDC* (see Figure 7).

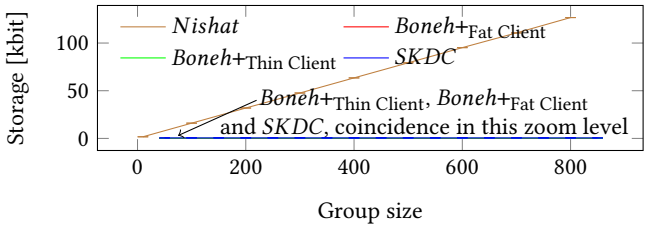

Figure 6: Temporary storage requirements for CI during group creation.

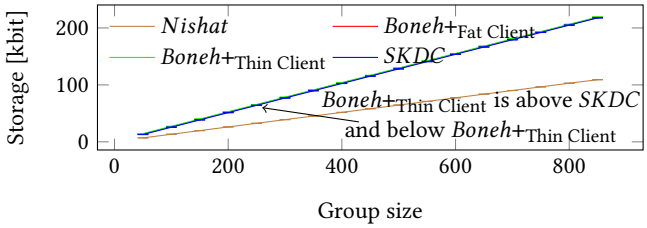

Figure 7: Permanent storage requirements for CI during group creation.

5.3 Discussion

We analyze the considered schemes in terms of metrics computation time, energy efficiency, and memory consumption and leave out their features and requirements for our first observation. With this pure focus on the performance metrics, we derive two findings from the evaluations. First, the workload distribution of a group encryption scheme can differ between the involved actors or phases. For example, *Nishat* has, concerning the group creation workload of the CI, the most extensive preparatory phase in terms of computation time (see Figure 2). This effort pays off for the CI in the operational phase, where *Nishat* is more efficient than the other schemes (see Figure 3). Figures 3 and 4 show how to vary the workload distribution between the actors. In contrast to *Boneh+*$_{\text{Fat Client}}$, *Boneh+*$_{\text{Thin Client}}$ distributes more load to the CI in the group creation process's operational phase, thus relieving the group members in this phase. Such information about the distribution of workloads across actors and phases provides valuable insight for developers to decide how to distribute the load across actors and phases for their specific use case.

The second observation is that none of the schemes presented performs best in all considered operations. However, there is a tendency for the baseline *SKDC* to be the best suitable scheme. However, this should not lead to the conclusion that *SKDC* should be the default in case of doubt. *SKDC* has corresponding requirements to work as intended and offer the appropriate performance. Thus *SKDC* sends individualized messages to its n group members, which would impose a corresponding overhead in a broadcast environment. In this case, group members would not receive just one message, but n, from which they would have to pick their message. Therefore, *SKDC* is not suitable for every use case, and the choice of the group encryption method should, therefore, rely on a benchmark that considers the schemes' requirements.

In summary, our observations emphasize that a benchmark for centralized group encryption schemes must consider the performance of the schemes and their requirements and features.

6 THREATS TO VALIDITY

In this paper, we focus on specifying a benchmark for centralized group encryption schemes for IoT. However, our benchmark relies on assumptions regarding the communication between group members and the CI, which could pose a potential threat to our benchmark's validity. For this reason, we discuss these assumptions and their possible impact on our benchmark in more detail below. In simple terms, we summarize the assumptions as follows: We assume that communication between the CI and the group members takes place instant and undisturbed. Thereby, it does not matter whether possible disturbances originate from an attacker or occur coincidentally by faulty message transmissions. Possible attack vectors would be to (i) delay, (ii) block, (iii) retransmit, or (iv) modify messages. However, these problems generally apply to all messages sent in a distributed IoT scenario and not only in group encryption schemes. For this reason, we see this as an independent problem detached from the group encryption scheme and as an orthogonal research area. Since we also want to benchmark the pure group encryption method and not the reliability of message delivery in distributed systems, we do not see any limitation of the validity of our contribution by our assumptions.

7 RELATED WORK

In this section, we discuss relevant related work and highlight the novelty of our contributions.

First, literature provides approaches that compare group encryption schemes with each other (e.g. [8, 16]). However, these approaches only use theoretical analysis or use different metrics or different definitions of similar metrics. Concerning metrics, the main focus lies on calculation times, neglecting, for example, temporary or permanent storage requirements. These works also focus

on specific workloads of certain actors and do not consider the complete workload of all involved actors and the workload distribution between different phases. These comparisons do not consider the features and requirements of the schemes.

Another stream of research considers practical analysis and comparisons based on measurements or simulations in the literature. For example, some works [1, 3] focus on analyzing the encryption and decryption process but do neither discuss nor analyze how to generate and distribute the required keys and do not consider requirements and features. While other works (e.g. [2, 27]) focus purely on the required transmission time or the number of key agreement messages, they do not differentiate between different actors and phases. Furthermore, the respective practical works use their measurement environment without considering the measuring instruments' accuracy when determining the metrics.

In summary, related work focuses on the theoretical or practical analysis of certain aspects of group encryption methods, such as the time required to encrypt and decrypt a message to the group after establishing a group key. Thus, the used metrics and their definitions hamper a direct comparison. They consider neither the workload of all involved actors nor the division of the workload into different phases. Furthermore, the literature does not consider the scheme's requirements and features for comparison and performs practical performance measurements on different hardware or gives incomplete testbed descriptions. In this work, we contribute to those issues.

8 CONCLUSION & FUTURE WORK

In this paper, we presented a benchmark for centralized group encryption. In contrast to related work, we target a holistic approach: Besides the performance analysis, we provide a feature-driven analysis that considers schemes' requirements and features. Using the same hardware in each case is essential to determine the schemes' metrics comparably. Therefore, we offer a standardized test environment with our measurement setup, which also considers the measuring instruments' accuracy. Also, we want to consider, by default, the complete workload of a group encryption scheme, broken down by phase and actors, and all metrics, as presented in this paper, and not just a subset of them. The evaluation shows, no method performs best in every analyzed dimension. Accordingly, developers need to determine their exact requirements for the communication aspect and choose the scheme that fits those application requirements best.

With this work, we performed the first step towards a generic benchmark for group encryption schemes. Next, we plan a systematic literature review to provide a complete list of group encryption schemes' requirements and features. Further, our benchmark currently focuses on centralized schemes; after extending the benchmark to support distributed schemes, we can extend the set of analyzed group encryption schemes. This ability will generate an extensive knowledge base containing both information from the theoretical analysis and the performance benchmark to support developers choosing a group encryption scheme for their applications. Finally, the knowledge base can be the foundation for a self-adaptive [18] or self-aware computing system, which can switch the group encryption scheme at runtime depending on the system environment, the system characteristics, and goals. This foundation would also complement our previous work in design patterns for IoT systems with security-related aspects [19].

ACKNOWLEDGMENTS

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) [25].

REFERENCES

[1] O.O. Adekanmbi et al. 2015. Performance Evaluation of Common Encryption Algorithms for Throughput and Energy Consumption of a Wireless System. *JOURNAL OF ADVANCEMENT IN ENGINEERING AND TECHNOLOGY* (06 2015).
[2] Y. Amir et al. 2002. On the performance of group key agreement protocols. In *Proceedings 22nd International Conference on Distributed Computing Systems.*
[3] Yair Amir et al. 2004. On the performance of group key agreement protocols. *ACM Transactions on Information and System Security (TISSEC)* 7, 3 (2004), 457–488.
[4] Xirong Bao et al. 2015. A key management scheme based on grouping within cluster. *WCICA* 2015 (03 2015). https://doi.org/10.1109/WCICA.2014.7053290
[5] S. Bipasha Biswas et al. 2018. Solar Water Pumping System Control Using a Low Cost ESP32 Microcontroller. In *CCECE.*
[6] Dan Boneh et al. 2005. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Advances in Cryptology – CRYPTO 2005.*
[7] Yacine Challal et al. 2008. Group Key Management Protocols: A Novel Taxonomy. *International Journal of Computer and Information Engineering* 2, 10 (2008).
[8] Omar Cheikhrouhou. 2015. Secure Group Communication in Wireless Sensor Networks: A survey. *Journal of Network and Computer Applications* 61 (11 2015).
[9] Ling Cheung et al. 2007. Collusion-Resistant Group Key Management Using Attribute-Based Encryption. *IACR Cryptology ePrint Archive* 2007 (01 2007), 161.
[10] Yokogawa Test & Measurement Corporation. [n.d.]. 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.
[11] Free Software Foundation. [n.d.]. The GNU Multiple Precision Arithmetic Library. Online available under https://gmplib.org/, Accessed on 24.06.2020.
[12] OpenSSL Software Foundation. [n.d.]. OpenSSL Cryptography and SSL/TLS Toolkit. Online available under https://www.openssl.org/docs/man1.1.0/man1/openssl-enc.html, Accessed on 10.10.2020.
[13] Romain Gay et al. 2018. Tight Adaptively Secure Broadcast Encryption with Short Ciphertexts and Keys. In *Security and Cryptography for Networks.*
[14] CNSS Glossary Working Group et al. 2006. National Information Assurance (IA) Glossary. *CNSS Instruction No. 4009* (2006).
[15] Mark Hung. 2017. Leading the IoT - Gartner Insights on How to Lead in a Connected World. Online available under https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf, Accessed on 24.01.2020.
[16] Bibo Jiang et al. 2008. A survey of group key management. In *2008 international conference on computer science and software engineering*, Vol. 3. IEEE, 994–1002.
[17] Firdous Kausar et al. 2007. Secure group communication with self-healing and rekeying in wireless sensor networks. In *International Conference on Mobile Ad-Hoc and Sensor Networks.* Springer, 737–748.
[18] Christian Krupitzer et al. 2015. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* 17 (2015), 184–206.
[19] Christian Krupitzer et al. 2020. An Overview of Design Patterns for Self-Adaptive Systems in the Context of the Internet of Things. *IEEE Access* 8 (2020).
[20] Veronika Lesch et al. 2021. A Comparison of Mechanisms for Compensating Negative Impacts of System Integration. *Future Generation Computer Systems* (March 2021).
[21] Shu-Quan Li et al. 2010. A survey on key management for multicast. In *2010 Second International Conference on Information Technology and Computer Science.*
[22] Ben Lynn. [n.d.]. The Pairing-Based Cryptography Library. Online available under https://crypto.stanford.edu/pbc/, Accessed on 24.06.2020.
[23] Koti Nishat et al. 2016. Group-oriented encryption for dynamic groups with constant rekeying cost. *Security and Communication Networks* 9, 17 (2016).
[24] Thomas Prantl et al. 2020. Evaluating the Performance of a State-of-the-Art Group-oriented Encryption Scheme for Dynamic Groups in an IoT Scenario. In *MASCOTS (MASCOTS '20).*
[25] Thomas Prantl et al. 2020. SIMPL: Secure IoT Management Platform. In *ITSec (1st ITG Workshop on IT Security).*
[26] S.P.E.C. 2014. Power and Performance Benchmark Methodology V2.2.
[27] Shanyu Zheng et al. 2005. Performance of Group Key Agreement Protocols over Multiple Operations.. In *IASTED PDCS.* Citeseer, 600–606.