Analytical Models for the Scalability of Dynamic Group-key Agreement Protocols and Secure File Sharing Systems

GOKCAN CANTALI, Dept. of Computer Engineering, Bogazici University
ORHAN ERMIS, Dept. of Computer Engineering, Bogazici University and EURECOM
MEHMET UFUK ÇAĞLAYAN, Dept. of Computer Engineering, Yasar University
CEM ERSOY, Dept. of Computer Engineering, Bogazici University

Dynamic group key agreement protocols are cryptographic primitives to provide secure group communications in decentralized and dynamic networks. Such protocols provide additional operations to update the group key while adding new participants into the group and removing existing participants from the group without re-executing the protocol from the beginning. However, the lack of scalability emerges as one of the most significant issues of dynamic group key agreement protocols when the number of participants in the group increases. For instance, frequent participant join requests for large groups may cause an effect similar to a Distributed Denial of Service (DDoS) attack and violate the system availability due to the increase in group key update time. Therefore, analyzing the scalability of dynamic group key agreement protocols is crucial to detect conditions where the system becomes unavailable. In this article, we propose an analytical performance model to evaluate the scalability of dynamic group key agreement protocols by using queueing models. We also extend our performance model for evaluating the scalability of secure file sharing systems that utilize group key agreement protocols. Moreover, we present a demonstrative use case to show the applicability of our performance model on an example group key agreement protocol and a secure file sharing system.

CCS Concepts: • **Security and privacy** → **Key management**; • **Networks** → **Network performance evaluation**; • **Information systems** → *Information storage systems*;

Additional Key Words and Phrases: Performance Model, Dynamic Group-Key

ACM Reference format:
Gokcan Cantali, Orhan Ermis, Mehmet Ufuk Çağlayan, and Cem Ersoy. 2019. Analytical Models for the Scalability of Dynamic Group-key Agreement Protocols and Secure File Sharing Systems. *ACM Trans. Priv. Secur.* 22, 4, Article 20 (September 2019), 36 pages.
https://doi.org/10.1145/3342998

1 INTRODUCTION

Group key agreement protocols are instrumental to establish a secure communication for a set of participants. In such protocols, the security of a communication is accomplished via using a

This work is supported by the Turkish Ministry of Development under the TAM Project number DPT2007K120610.
Authors' addresses: G. Cantali and C. Ersoy, Dept. of Computer Engineering, Bogazici University, Istanbul, Turkey, 34342; emails: {gokcan.cantali, ersoy}@boun.edu.tr; O. Ermis, Dept. of Computer Engineering, Bogazici University and EURECOM, Biot, Sophia-Antipolis, France, 06410; email: ermis@eurecom.fr; M. Ufuk Çağlayan, Dept. of Computer Engineering, Yasar University, Izmir, Turkey, 35100; email: ufuk.caglayan@yasar.edu.tr.

2471-2566/2019/09-ART20 $15.00
https://doi.org/10.1145/3342998

common key computed by all cooperating participants in the group. Diffie-Hellman Key Exchange Protocol is the first key agreement protocol that operates for a group of two participants. Then, the idea was extended to multiple participants by Ingemarsson et al. in Reference [25]. Following the multiple participant setup, two important group key agreement protocols with and without authentication were proposed by Burmester and Desmedt in Reference [7]. Authentication property is used for confirming the identities of participants in group communications. Another significant security property is the fault-tolerance property [43], which is used for detecting and removing the malicious participant from the group key computation. Forward secrecy property [14] also plays an important role for the security of group key agreement protocols. The property was adopted by Tseng [40] for protecting against the compromise of former and subsequent group keys of a protocol, if the long-term key of a participant is compromised.

Former group key agreement protocols mostly operate on static groups [23, 42, 49], in which the set of participants remains unchanged till the end of a communication session or the protocol is re-executed from the beginning when the set of participants is altered. However, recent trends in communication technologies require dynamic settings for the set of participants due to the significant overhead of updating the group key. Accordingly, group key agreement protocols have evolved to overcome such overhead by providing dynamic group operations [11, 19, 41]. The most common dynamic group operations are the join of new participants into the group and the leave of existing participants from the group. Such join and leave operations are sometimes called auxiliary group key agreement operations. In contrast to static group key agreement protocols, Dynamic Group Key Agreement Protocols (DGKAPs) support auxiliary operations by selecting a small subset of participants as active participants. Active participants, which are entities in the group, are responsible for updating the group key by re-executing the protocol from the beginning after any join or leave has occurred. Since the set of active participants is a smaller group than the original set, dynamic group key agreement protocols provide better performance than static ones.

Performance analysis of DGKAPs is a significant issue for evaluating the applicability of a protocol in a real-life application. However, performance analysis of well-known DGKAPs only consider numerical methods and simulation techniques, and to the best of our knowledge, there is no analytical performance model for scalability analysis of DGKAPs. Therefore, it is not possible to measure effects of auxiliary operations on the group communication. For instance, if there exist frequent join requests to the group then, group members spend most of their time to update the group key rather than communicating with each other. Consequently, such requests may result in an effect like a Distributed Denial of Service (DDoS) attack that violates the availability of the group communication. Provision of an extensive scalability analysis by employing queueing models is expected to help learn availability boundaries of the group communication, which is our main motivation for the study. In addition, by using these models, we aim to address the following issues: (i) the average waiting time of a joining participant before joining the group, (ii) the effects of frequent participant arrivals on the availability of group communication, (iii) the interdependence between the average waiting time of a joining participant and the number of participants in the group.

Our main contributions in this article are as follows:

- We propose an analytical performance model on DGKAPs by using queueing models to analyze the scalability of protocols. Our performance model can be used for detecting the conditions where the system availability is violated.
- We extend our performance model for secure file sharing systems that utilize group key agreement protocols as a file confidentiality service.

- Moreover, we model secure file sharing operations such as file download/upload and file encryption/decryption by using queueing models.
- We illustrate the applicability of our model on different group key agreement protocols in References [18, 19, 26, 48].
- We also present a demonstrative use case of our model on Private File Sharing System (PFSS) [19], and we present numerical performance results of the system.

The rest of the article is organized as follows: In Section 2, we provide a comparison of performance analysis methods used in secure multi-party communication systems. Section 3 explains the proposed queueing-based performance model. In Section 4, application of our model on different dynamic group key agreement protocols [18, 19, 26, 48] is given. Section 5 extends the proposed performance model to secure file sharing systems with a demonstrative use-case scenario. Section 6 concludes the article.

2 RELATED WORK

In this section, we first provide preliminary material on queueing theory. Then, we present a comparison of performance evaluation methods used in different secure multi-party group communication systems. Finally, we discuss how dynamic group key agreement protocols benefit from an analytical performance model.

2.1 Preliminaries for Queueing Theory

Queueing Theory [37] is a mathematical model to analyze the number of participants in the queue and the waiting time of participants in the queue by considering the nature of a specific application. A queue can be formally represented by using the Kendall's Notation as given in the following definition:

Definition 2.1. Let Q be an $X/Y/Z$ queue. Then, the distributions of X, Y, and Z can be expressed as follows:

- X represents the distribution of participants' arrival time and denoted as follows:

$$X := \begin{cases} M & \text{if the inter-arrival time is exponentially distributed} \\ D & \text{if the inter-arrival time is pre-determined} \\ G & \text{if the inter-arrival time is arbitrarily distributed.} \end{cases}$$

- Y represents the distribution of participants' service time and denoted as follows:

$$Y := \begin{cases} M & \text{if the service time is exponentially distributed} \\ D & \text{if the service time is pre-determined} \\ G & \text{if the service time is arbitrarily distributed.} \end{cases}$$

- Z represents the number of simultaneously working servers. It is assumed that one server can only serve one participant at a time.

Definition 2.2. Let Q be an $X/Y/Z$ queue. We use the following input parameters as defined in Reference [37]:

- **Participant Arrival Rate (λ)** represents the average number of arriving participants into the system per unit time.
- **Service Rate (μ)** represents the mean number of participants that are served per unit time.

- **Mean Service Time ($E[S]$)** represents the average time a participant spends in the server and it is computed as follows:
$$E[S] = \frac{1}{\mu}.$$
- **Server Utilization (ρ)** The server utilization represents the fraction of time in which a server is busy. The server utilization can be expressed as:
$$\rho = \frac{\lambda}{Z\mu}.$$
- **Arrival Rate Variance (σ_a^2)** is the variance of the inter-arrival time distribution and it is zero if arrivals are deterministic:
$$X = D \Rightarrow \sigma_a^2 = 0.$$
- **Service Time Variance (σ_s^2)** is the variance of the service time distribution and it is zero if the service durations are deterministic:
$$Y = D \Rightarrow \sigma_s^2 = 0.$$

The main performance metrics used in queueing theory are the average queue length and the average waiting time. The list of widely used performance metrics are given as follows:

Definition 2.3. Let Q be a $X/Y/Z$ queue. The list of performance metrics of Q are given as follows:

- The Average Participant Waiting Time (W): In the long run of a queueing simulation, the average participant waiting time represents how much time, on average, a participant has to wait in the system before the participant is served. W includes both the time spent in the queue and the time spent in the server.
- The Average Participant Waiting Time in Queue (W_q): In the long run of a queueing simulation, W_q represents how much time, on average, a participant has to wait in the queue before the participant is served. The relation between W and W_q can be expressed as follows:
$$W = W_q + E[S].$$
- The Average Number of Participants (L): The average number of participants represents how many participants are expected to be in the system in the long run of a simulation. L includes both the number of participants in the system and the number of participants in the queue.
- The Average Number of Participants in Queue (L_q): The average number of participants in queue represents how many participants are expected to be in the queue in the long run of a simulation. The relation between L and L_q can be expressed as follows:
$$L = L_q + \lambda E[S].$$

2.2 A Comparison of Performance Evaluation Methods Used in Secure Multi-Party Group Communication Systems

While comparing DGKAPs and secure file sharing systems, we use the following comparison criteria:

- **Asymptotic Complexity:** We consider asymptotic complexity analysis as a derivation of an upper bound regarding time complexity of a system by using O notation.
- **Numerical Methods:** We consider numerical methods as a subset of asymptotic complexity analysis. Numerical analysis is performed when time complexity of the system is estimated for a fixed number of participants but not derived asymptotically by using O notation.

Table 1. Performance Evaluation Methods Used in Well-known Group Key Agreement Protocols

	Numerical Methods	Asymptotic Complexity	Simulation	Analytical Performance Model
Abdel-Harfez et al. [2]	✓	✗	✓	✗
Augot et al. [4]	✗	✗	✓	✗
Change et al. [8]	✓	✗	✓	✗
Chang et al. [10]	✓	✓	✓	✗
Cheng et al. [11]	✓	✗	✓	✗
Chung [13]	✓	✗	✗	✗
KAP-PBC [16]	✓	✓	✓	✗
GKAP-MANET [18]	✓	✓	✓	✗
Gangwar et al. [20]	✓	✓	✗	✗
Huang et al. [23]	✓	✓	✗	✗
Li et al. [27]	✓	✗	✓	✗
Teng et al. [39]	✓	✗	✗	✗
Tzeng [42]	✓	✗	✗	✗
Tzeng [43]	✓	✗	✗	✗
Zhang et al. [48]	✓	✓	✗	✗
Zhao et al. [49]	✗	✗	✓	✗

- **Simulation:** Performance analysis is performed via simulation when the system runs on a simulation environment for different ranges of parameters.
- **Analytical Performance Model:** We consider networks of queues as analytical performance models. If the performance of a communication system is analyzed by using a queueing model, we state that analytical performance models are used.

Methods used for performance evaluation of well-known DGKAPs are as given in Table 1. Comparisons show that numerical methods are the most widely used by the protocols. Simulation is the second most common method, and asymptotic complexity analysis is less widely utilized. Note that only three protocols [10, 16, 18] use both simulation and asymptotic complexity analysis simultaneously. However, none of the example DGKAPs have been analyzed by using analytical performance models. Although the study in Reference [17] compares four different DGKAPs with respect to scalability of protocols, it does not provide any analytical performance model.

DGKAPs are also utilized in secure file sharing systems (SFSSs) as a file confidentiality service by using the computed group key to encrypt and decrypt shared files. Methods used for performance analysis of some well-known SFSSs are as given in Table 2. According to the comparison, all SFSSs use numerical methods for evaluating their performance. HABE [45], which is a combination of hierarchical identity-based encryption [6] and ciphertext-policy attribute-based encryption [46], uses proxy re-encryption [1] and lazy revocation [5] to improve performance of participant removal operation. However, the study does not provide simulation results or analytical performance models. None of the example SFSSs presented here use analytical performance models.

In this work, our main goal is to use analytical performance model while addressing the scalability issues of DGKAPs, since such models have more advantages over other performance evaluation methods. For instance, asymptotic complexity analysis yields results indicating the relation between the number of participants in the group and the time required to complete a join operation. However, in DGKAPs new participants may keep coming and request to join the group. In such case, the time required to complete a join operation increases as the number of participants

Table 2. Performance Evaluation Methods Used in Well-known Secure File Sharing Systems

	Numerical Methods	Asymptotic Complexity	Simulation	Analytical Perfomance Model
System in [12]	✓	✓	✓	✗
PFSS [19]	✓	✓	✓	✗
System in [22]	✓	✓	✗	✗
mCP-ABE [24]	✓	✗	✗	✗
TimePRE [28]	✓	✓	✓	✗
Mona [29]	✓	✓	✓	✗
System in [30]	✓	✗	✓	✗
FADE [38]	✓	✗	✗	✗
HABE [45]	✓	✓	✗	✗
System in [47]	✓	✗	✗	✗
System in [50]	✓	✗	✓	✗

in the group increases. However, it is difficult to observe the difference between performances of consecutive join and/or leave operations in asymptotic complexity analysis and thereby it is also difficult to observe the scalability of the protocol. Numerical evaluation can be defined as an alternate approach for analytical performance model with more accurate results. However, we can only make estimations for a fixed number of participants and for a fixed time period, but it is not possible to obtain detailed estimations for a continuous time period.

Similar to the asymptotic complexity analysis, simulation techniques that are used in previous studies do not provide general results for all possible scenarios when the scalability of DGKAPs are concerned. Although they provide almost realistic performance results, simulations may need to be executed for all possible cases such as different arrival rates of participants or varying waiting time of participants, and so on. One important example for both asymptotic complexity analysis and simulation-based performance analysis is the work proposed in Reference [3]. The proposed work provides a comprehensive performance evaluation of five different DGKAPs and gives a comparison on their performance results based on both asymptotic analysis and simulations. Nevertheless, the study does not consider the long-term execution of protocols in terms of varying residence times of participants in the group, participant leaving and joining rates, or waiting time before joining the group. For this respect, we propose an analytical performance model that employs queueing theory to analyze the scalability of DGKAPs. Moreover, to the best of our knowledge, queueing theory is first used in this work for the analysis of DGKAPs.

3 PERFORMANCE MODEL FOR DYNAMIC GROUP KEY AGREEMENT PROTOCOLS

In this section, we provide details about our performance model and explain how we model leave and single/mass join operations of DGKAPs. For this respect, we analyze the scalability of DGKAPs in terms of three different performance models, namely Residence Model, Join Model, and Leave Model.

An overall view of the proposed analytical performance model is shown in Figure 1. Since the number of participants in the group is the most important information while analyzing the performance of DGKAPs, we first need to model the participants in the group. Therefore, we use Residence Model to formulate the expected number of participants in the group and residence time of participants in the group. Then, we use the expected number of participants to model mass/single join and leave operations. In our analytical performance model, we assume that the

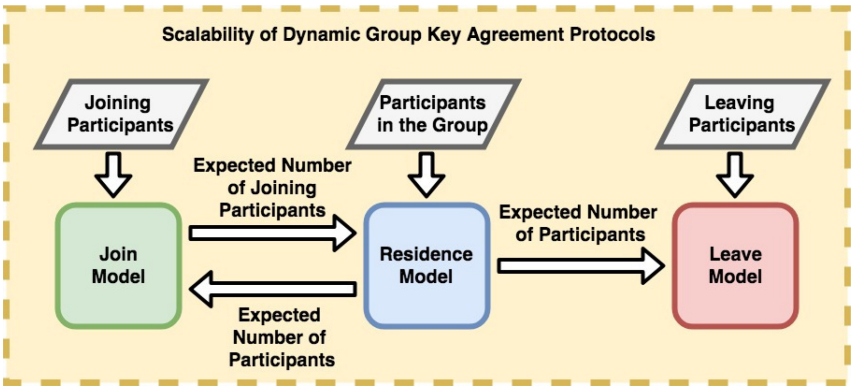

Fig. 1. An overall view of the proposed performance model for dynamic group key agreement protocols.

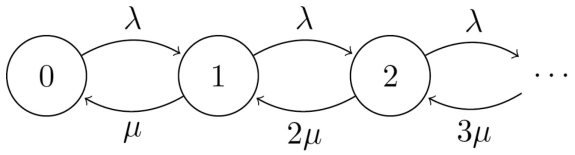

Fig. 2. The steady-state representation of an $M/M/\infty$ queue.

number participants in the group affects join operation, in particular Join Model, and the number of joining participants affects Residence Model. Thus, Join Model and Residence Model are dependent on each other. However, we also assume that the number of participants in the group affects leave operation, therefore, Leave Model is also dependent on Residence Model. However, residence time of each participant is defined by Residence Model not Leave Model, which makes Residence Model independent from the Leave Model.

The number of participants in the group significantly affects the time required to perform the group key computation. We consider the number of participants in the group as a parameter that is represented with an infinite server queueing model, namely the residence model. In the residence model, the service time is considered as the residence time of a participant in the group until participants leave the group. Each participant in the group is served by a different server. Hence, the number of busy servers at time t equals to the number of participants in the group at time t:

$$s_b(t) = k(t), \tag{1}$$

where s_b represents the number of busy servers and k represents the number of participants in the group.

We assume that inter-arrival time in the residence time model is exponentially distributed and the residence time of participants is arbitrarily distributed. Thus, we use an $M/G/\infty$ queue that represents a system with infinite servers, arbitrarily distributed service time, and exponentially distributed inter-arrival time. In this part of our analysis, we assume that each participant in the group is also modelled as a server in the model, since group key computation is realized by all cooperating group members. Therefore, in such circumstances, participants do not wait in the queue and begin to get served as soon as they arrive into the system.

The study in Reference [34] shows that steady-state probabilities of an $M/G/\infty$ queue are the same as an $M/M/\infty$ queue, and Figure 2 shows the steady-state representation of an $M/M/\infty$

queue. By using Figure 2, we derive the performance metrics of an $M/G/\infty$ as follows:

$$P_1 = \frac{\lambda}{\mu} P_0, \tag{2}$$

$$P_k = \frac{\lambda}{k\mu} P_{k-1}, \tag{3}$$

where k is the number of participants in the group, λ is the participant arrival rate, μ is the group residence rate, and P_k is the steady-state probability that k participants are in the group. Then, the closed-form expression for P_k (see Appendix A.2 for intermediate steps) is derived as

$$P_k = \frac{\left(\frac{\lambda}{\mu}\right)^k e^{-\frac{\lambda}{\mu}}}{k!}. \tag{4}$$

We observe that Equation (4) is the probability mass function of the Poisson distribution with rate λ/μ. Therefore, the number of participants in the group parameter is considered as a Poisson-distributed variable.

3.1 Analytical Model of a Single Join Operation

We model the single join operation by using a queueing model. When a participant requests to join the group, the request is considered as an arrival event into the queue. The completion of a join operation is considered as a departure event from the queue. To model the arrival of new participants for a dynamic group key agreement protocol (DGKAP), we should examine empirical arrival distributions for multi-party communication applications that can utilize DGKAPs. For instance, References [36, 44] show empirical results that the inter-arrival time for a live video streaming application is exponentially distributed. Since a DGKAP can be used in a live video streaming application, we assume that the inter-arrival time for a single join operation is exponentially distributed. We also assume that the service time is exponentially distributed for simplicity. Therefore, we use a single server queue with exponentially distributed inter-arrival and service time for the join model.

For estimating the number of participants in the group, we use the residence model. We connect the join model with the $M/G/\infty$ queue in the residence model as shown in Figure 3. The output of the join operation is supplied as the input into group residence model. Until we analyze the leave operation in Section 3.3, we ignore the leave model in Figure 3.

Burke's Theorem states that an $M/M/1$ queue with λ participant arrival rate produces an output with rate λ. Therefore, a queue that is sequentially connected to the output of an $M/M/1$ queue has exponentially distributed inter-arrival time. Since our join model consists of an $M/M/1$ queue, our residence model has exponentially distributed inter-arrival time.

When participants arrive and request to join the group, they stay in the join queue until they are ready to join. After a participant joins the group, the participant arrives at the group residence server and stays in the server until deciding to leave the group. In Figure 3, λ represents the participant arrival rate of joining participants, $1/\mu_{join}$ represents the service time for the join operation, and $1/\mu$ represents the mean group residence time as explained in Table 3.

We now analyze the performance of the join operation by using both join and group residence models. In our model, μ_{join} depends on the number of participants in the group, which is Poisson-distributed with rate λ/μ as derived in Equation (4). Thus, the expected value of the number of participants in the group, k, equals to

$$E[k] = \frac{\lambda}{\mu}. \tag{5}$$

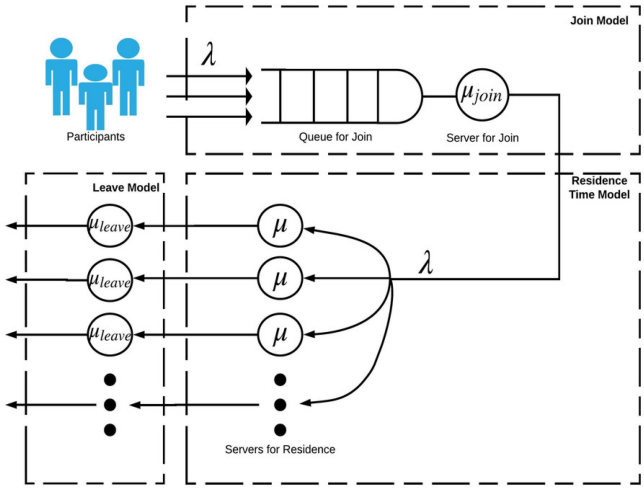

Fig. 3. Overview of our performance model.

Table 3. Definition of the Parameters Used in the Proposed Model

Parameter	Explanation
λ	The participant arrival rate of new participants
μ_{join}	The service rate of join operation
$E[S_{join}] = \frac{1}{\mu_{join}}$	The mean service time of join operation
μ	The group residence rate
$E[S_{res}] = \frac{1}{\mu}$	The mean group residence time
S^k_{join}	The service time for a new participant to join the group when there are k participants in the group
W	The average waiting time of a new participant before joining
L	The average number of participants waiting for join operation at an arbitrary time
k	The number of existing participants in the group
p_k	The probability that there are k existing participants in the group
s_b	The number of busy servers in the group residence model

We approximate the mean service time of the join operation by fixing the number of participants in the group to the expression in Equation (5):

$$E[S_{join}] \approx \frac{E\left[S_{join}^{\lceil \frac{\lambda}{\mu} \rceil}\right] + E\left[S_{join}^{\lfloor \frac{\lambda}{\mu} \rfloor}\right]}{2}. \tag{6}$$

Then, the average waiting time of a joining participant, W, is derived (see Appendix A.1 for intermediate steps) as

$$W = \frac{1}{\frac{1}{E[S_{join}]} - \lambda} \tag{7}$$

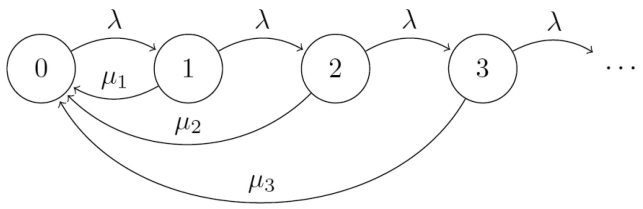

Fig. 4. The steady-state representation of the mass join operation.

and the average number of participants who are waiting to join the group, L, is derived (see Appendix A.1 for intermediate steps) as

$$L = \frac{\lambda}{\frac{1}{E[S_{join}]} - \lambda}. \tag{8}$$

3.2 Analytical Model of a Mass Join Operation

Mass join operation allows multiple participants to simultaneously join the group such as protocols in References [16, 19]. During the modeling of the mass join operation, we use batch queues [9]. We assume that both inter-arrival time and service time of the mass join operation is exponentially distributed. We also assume that new participants arrive one-by-one but join in bulks. Therefore, we use a single server queue with batch service to represent the mass join operation.

The batch size of the join service, Y, is not constant, since it depends on the number of participants waiting in the queue to join the group. In general, Y in bulk service queues is either fixed or considered as a random variable that is smaller than the queue size [33]. However, we assume that the system always allows all participants in the queue to join the group. Therefore, Y is equal to the number of participants waiting in the queue.

Figure 4 shows the steady-state probabilities in the mass join operation. Since the batch size is considered the same as the queue size, each state goes back to the empty state after a departure event. In the mass join operation, the number of joining participants may affect the service time of the operation. Each state has its own service time denoted as $\mu_1, \mu_2 \ldots$ We derive the balance equations as

$$P_k \lambda = P_{k+1}(\lambda + \mu_{k+1}). \tag{9}$$

By using Equation (9), the closed-form expression of P_0 is derived as (see Appendix A.2 for intermediate steps):

$$P_0 = \left[1 + \sum_{k=1}^{\infty} \left(\frac{\lambda^k}{\prod_{i=1}^{k} (\lambda + \mu_i)} \right) \right]^{-1}. \tag{10}$$

The closed-form of P_0 given in Equation (10) can be applied to a dynamic group key agreement protocol that supports the mass join operation. However, deriving concrete performance results from such a complicated equation is difficult. Instead, we follow another approach to derive an upper bound for the performance metrics.

Mass join operation allows every participant who waits in the queue to join the group. The waiting queue is emptied when a mass join operation starts. If new participants arrive during a mass join operation, they only wait in the queue until the next operation starts. Therefore, the average waiting time in the queue, W_q, is less than or equal to the mean service time of the mass join operation, $E[S]$. By using $W_q \leq E[S]$, we can derive an upper bound for W and L. The process of upper bound derivation for mass join operation is thoroughly explained in Section 4.

Table 4. Definition of the Parameters Used in the Leave Model

Parameter	Explanation
$E[S_{res}]$	The mean residence time of participants
$E[S_{leave}]$	The mean service time of leave operation
$E[S_{totalres}] = E[S_{res}] + E[S_{leave}]$	The total residence time that is the sum of the mean residence time and the mean service time of leave operation
$\mu = 1/E[S_{res}]$	The group residence rate
$\mu_{leave} = 1/E[S_{leave}]$	The service rate for leave operation
$\mu_{total} = 1/E[S_{totalres}]$	The total residence rate

3.3 Analytical Model of a Leave Operation

In group key agreement protocols, participants should not be able to take part in the group communication after they leave the group. The group key needs to be updated for preventing the leaving participants to access the encrypted communication. Therefore, most dynamic group key agreement protocols provide a leave operation to update the group key after a participant leave occurs. We now propose a model for the scalability analysis of a leave operation by adding a third system as an output to the group residence model. When a participant requests to leave the group, they are removed from the group residence model and added into the leave model where they wait until the group key update process is completed. When the group key is updated, the leaving participant is removed from the leave model.

In some group key agreement protocols [18, 19], multiple participants can leave the group simultaneously without waiting in a queue. A participant does not depend on every participant in the group to leave. Instead, each participant, P_i, depends on a consecutive participant to perform the necessary computations and communications. In the context of this article, two participants, P_i and P_{i-1}, are called consecutive participants if their indices differ by one. For some protocols, participant list is circular, where $P_{n+i} = P_i$ and n is the number of participants in the group. For the leave model, we assume that, in general, the leave of participant P_i is handled by its consecutive participants P_{i-1} and P_{i-2}. In this study, we also assume that the likelihood of consecutive participants leaving at the same time is considerably low. By using these assumptions, we model a leave operation with parallel servers as shown in Figure 3 and the model parameters are given in Table 4.

We sequentially connect residence servers and leave servers to derive the total residence time:

$$E[S_{totalres}] = E[S_{res}] + E[S_{leave}], \tag{11}$$

which implies

$$\frac{1}{\mu_{total}} = \frac{1}{\mu} + \frac{1}{\mu_{leave}}, \tag{12}$$

$$\mu_{total} = \frac{\mu.\mu_{leave}}{\mu + \mu_{leave}}, \tag{13}$$

where μ is the residence rate and μ_{leave} is the service rate for a leave operation. By using this approach, we remove the leave queue and extend the scope of residence model to include a leave operation. Accordingly, the steady-state probability distributions of the residence model need to be modified. Equation (4) is modified to

$$P_k = \frac{\left(\frac{\lambda}{\mu_{total}}\right)^k . e^{-\frac{\lambda}{\mu_{total}}}}{k!}, \tag{14}$$

Table 5. A General Terminology for DGKAPs

Term	Explanation
DGKAP operation	An operation provided by a dynamic group key agreement protocol, such as join, leave, mass join, and mass leave.
DGKAP function	A function that is used during one of the DGKAP operations. For instance, **Group Key Computation and Session Key Verification** functions are used while executing the join operation of KAP-PBC.
Mathematical operation	An operation that is used in any DGKAP function such as **modular exponentiation** and **bilinear mapping**, and so on.
Symmetric DGKAP	A DGKAP is symmetric if the computed group key is used to both encrypt and decrypt the messages. For instance, KAP-PBC in Reference [19] is a symmetric DGKAP.
Asymmetric DGKAP	A DGKAP is asymmetric if the group key is used only to encrypt the messages and each participant uses unique decryption keys to decrypt the messages. For instance, IBAAGKAP in Reference [48] is an asymmetric DGKAP.
Round	The time duration of a DGKAP operation in which participants perform a set of computations and communications.

where μ_{total} is defined in Equation (13). Moreover, we can extend the expected service time of join operation as follows:

$$E[S_{join}] = \frac{E\left[S_{join}^{\lceil \frac{\lambda}{\mu_{total}} \rceil}\right] + E\left[S_{join}^{\lfloor \frac{\lambda}{\mu_{total}} \rfloor}\right]}{2}. \tag{15}$$

Updated $E[S_{join}]$ is used for the derivation of performance metrics in Equations (7) and (8).

4 ANALYSIS OF DYNAMIC GROUP KEY AGREEMENT PROTOCOLS BY USING THE PROPOSED PERFORMANCE MODEL

In this section, we explain the process of evaluating the performance of Dynamic Group Key Agreement Protocols (DGKAPs) by using the proposed model in Section 3. First, we provide details on computing the expected service time of a DGKAP operation, $E[S]$, and then present how our model can be applied on different DGKAPs in References [18, 19, 26, 48] by using the computed $E[S]$ of each protocol operation. Since every DGKAP has its own notation and terminology, we also present general notations as given below and a common terminology as shown in Table 5:

- $\mathcal{P}$: The set of participants in the group;
- $\mathcal{P}_A$: The set of active participants in the group who perform the computations and communications during the DGKAP operation, where $\mathcal{P}_A \subseteq \mathcal{P}$;
- n: The number of participants in the group. $n = |\mathcal{P}|$;
- R: The number of rounds in the DGKAP operation;
- T_p^r: Time spent by a participant p to complete computations and communications at round r, where $1 \leq r \leq R$ and $p \in \mathcal{P}_A$;
- $\mathcal{F}^r$: The set of DGKAP functions that are used in round r of the DGKAP operation;
- t_f: The time required by a participant to execute DGKAP function f. t_f is considered constant for a specific f;

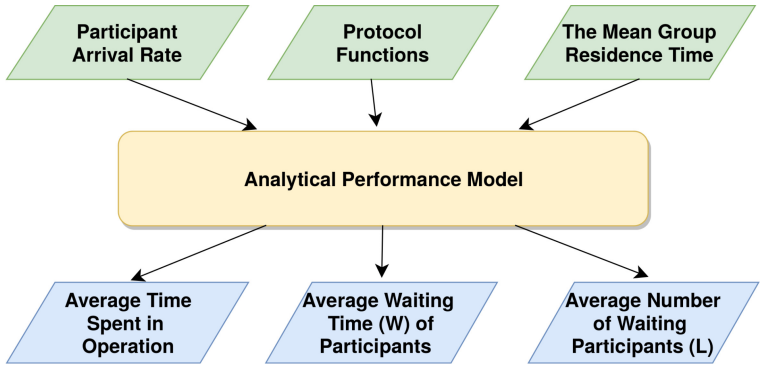

Fig. 5. Inputs and outputs of our performance model.

- $c_f^p(n)$: The number of times participant p executes a DGKAP function f during the DGKAP operation. c_f^p takes n as an input parameter, because the number of participants in the group affects the number of execution of a DGKAP function f.

While executing DGKAPs operations, a set of participants is selected as active participants $\mathcal{P}_A$ as defined in the notations above. Such participants are the ones that make more calculations than other participants in the group. Therefore, we first formulate time spent by active participants for each round r of DGKAP operation:

$$\forall p \in \mathcal{P}_A\ ,\ T_p^r = \sum_{f \in \mathcal{F}^r} c_f^p(n) * t_f\,. \tag{16}$$

Then, we consider the most time-spending participant among active ones for round r by computing the maximum of T_p^r values:

$$T_{max}^r = \max\left(\left\{T_p^r : p \in \mathcal{P}_A\right\}\right). \tag{17}$$

We assume that participants perform all computations and communications at the same time for each round. Therefore, round r takes at most T_{max}^r time to complete. We repeat the procedure in Equation (16) and Equation (17) for each round in the DGKAP operation, and then we obtain the following total execution time for the operation:

$$T_{max} = \sum_{r=1}^{R} T_{max}^r, \tag{18}$$

where T_{max} is the time required to execute the DGKAP operation. Hence, the mean service time $E[S]$ for DGKAP operation S is as follows:

$$E[S] = T_{max} = \sum_{r=1}^{R} T_{max}^r. \tag{19}$$

Throughout the article, we also assume that the participant arrival rate is λ and the group residence rate is μ. Given λ, μ, and the derived $E[S]$, we can compute the performance metrics for a DGKAP operation. In the rest of the analysis, we separately analyze join, mass join, and leave operations as shown in Figure 5. Our model takes average participant arrival rate, functions of the protocol to be analyzed, and average group residence time as input and outputs the average time spent in any operations of given DGKAP, average waiting time (W) of participants, and average number of waiting participants (L).

In the join operation, the right side of Equation (16) contains terms with variable n and those terms are transferred to $E[S]$ in Equation (19). We approximate the value of $E[S]$ as follows:

$$E[S] \approx E[S_{n \leftarrow E[n]}], \tag{20}$$

where $E[S_{n \leftarrow E[n]}]$ represents the mean service time of DGKAP operation when the value of variable n is fixed to $E[n]$.

Based on the results of our Residence Model in Section 3, the mean value of n is given by

$$E[n] = \frac{\lambda}{\mu} \tag{21}$$

for participant arrival rate λ and group residence rate μ. Therefore, we rewrite $E[n]$ as

$$E[n] = \frac{\lfloor \frac{\lambda}{\mu} \rfloor + \lceil \frac{\lambda}{\mu} \rceil}{2}. \tag{22}$$

We use floor function for $E[n]$, since the number of participants has to be integer value and this reduces to

$$E[n] = \left\lfloor \frac{\lambda}{\mu} \right\rfloor + 0.5. \tag{23}$$

By replacing the n terms in Equation (16) with the value of $E[n]$ in Equation (23), we obtain $E[S]$ with constant terms. Then, we obtain the performance metrics, W and L, by integrating $E[S]$ into the following equations:

$$W = \frac{1}{\frac{1}{E[S]} - \lambda}, \tag{24}$$

$$L = \frac{\lambda}{\frac{1}{E[S]} - \lambda}. \tag{25}$$

When we evaluate the performance of mass join operation of a DGKAP, we follow a different approach. First, $c_f^p(n)$ in Equation (16) becomes $c_f^p(n, m)$, where m represents the number of joining participants. Therefore, $E[S]$ in Equation (19) contains m terms in addition to n terms. There are two possible approaches to handle m terms.

In the first approach, m is considered constant. In this case, our analysis focuses on the performance of a DGKAP when there are always m joining participants. If m is a constant value, then the performance analysis of mass join operation is similar to the analysis of single join operation. Then, we can use Equations (20)–(25) to obtain the performance metrics.

In the second approach, m is considered as a random variable that changes for each mass join operation during the analysis. In this case, finding a closed-form expression for W and L is a significant issue. Instead, we derive an upper bound for W and L by using the following inequality:

$$W_q \leq E[S], \tag{26}$$

where W_q represents the average waiting time before the mass join operation starts and $E[S]$ is the mean service time of mass join operation. Participants who arrive during a mass join operation wait in the queue until the subsequent mass join operation starts. If a participant arrives at time τ after the operation is started, the participant waits $E[S] - \tau$. Therefore, for each case, Equation (26) holds. The subsequent operation joins all the participants waiting in the queue, and the expected number of joining participants, $E[m]$, is equal to the average number of waiting participants, L_q:

$$E[m] = L_q. \tag{27}$$

By using Little's Law, we re-write Equation (26) as

$$L_q \leq \lambda E[S]. \tag{28}$$

Then, we use a similar approximation as in the single join operation:

$$E[S] \approx E[S_{n \leftarrow E[n],\ m \leftarrow E[m]}], \tag{29}$$

where $E[S_{n \leftarrow E[n],\ m \leftarrow E[m]}]$ represents the mean service time when the value of n is fixed to $E[n] = \lambda/\mu$ and the value of m is fixed to $E[m] = L_q$. We integrate the value of approximated $E[S]$ into Equation (28) and obtain terms with L_q on both sides of the inequality. When we gather the L_q terms on the left side, we derive an upper bound as

$$L_q \leq \mathcal{E}(\ldots), \tag{30}$$

where $\mathcal{E}(\ldots)$ represents an expression related to the functions of the DGKAP operation and it changes with the protocol. By using Little's Law, we also obtain an upper bound for W_q as

$$W_q \leq \frac{1}{\lambda}\mathcal{E}(\ldots). \tag{31}$$

For the analysis of leave operation, active participants are also a proper subset of all participants in the group $\mathcal{P}_A \subset \mathcal{P}$. We make this assumption to ensure that multiple participants can leave the group simultaneously, without waiting for the leave of other participants. For instance, when participant P_i decides to leave, participants P_{i-1} and P_{i-2} may perform the necessary computations and communications to update the group key after P_i leaves. In this case, we call P_{i-1} and P_{i-2} consecutive participants of P_i. In general, the circular participant list is used in DGKAPs, where $P_{n+i} = P_i$ and n is the number of participants in the group. If a DGKAP provides a consecutive participant concept for leave operation, our model can be used to analyze the performance of leave operation. Otherwise, our model ignores the cost of leave operation and the participant is immediately removed from the system when their residence time is over.

Our analysis for the leave operation differs from the analysis of single join and mass join operations. Since we assume that participants can leave simultaneously and without waiting for others, there is no need for a queue to store leaving requests. We use an infinite server model similar to Residence Model and connect leave servers to residence servers, as shown in Figure 3 in Section 3.1. By taking the sum of the mean residence time and the mean service time of leave operation, we derive

$$E[S_{totalres}] = E[S_{res}] + E[S], \tag{32}$$

where $E[S]$ is the mean service time of leave operation, obtained the same way $E[S]$ of single join operation is obtained. Since there is no queue for leave operation, we cannot compute W or L for the leave operation. Instead, we consider the effects of leave operation on the performance metrics of join operation by updating W and L formulas of join operation. Equations (24) and (25) contain terms with $E[S_{res}]$. We replace such terms with $E[S_{totalres}]$ and re-calculate the performance metrics of join operation. If the DGKAP supports mass join operations, the same re-calculations are also made for the performance metrics of mass join operation.

We note that the analysis we have covered here focuses on the DGKAP functions instead of mathematical operations. Therefore, derived results are in terms of time costs of DGKAP functions. Our performance model also allows us to obtain the performance results of DGKAP operations in terms of mathematical operations. Such results can be derived with a slight modification of

Table 6. DGKAP Functions Used in KAP-PBC

Function	Parameter Symbol
Temporary Public Key Computation	t_{pkc}
Temporary Public Key Verification	t_{pkv}
Temporary Public Key Broadcast	t_{pkb}
Session Key Computation	t_{skc}
Session Key Verification	t_{skv}
Session Key Broadcast	t_{skb}
Group Key Computation	t_{gkc}

Equation (16) as follows:

$$\forall p \in \mathcal{P}_A \ , \ T_p^r = \sum_{f \in \mathcal{F}^r} c_f^p(n) \left(\sum_{o \in O^f} c_o^f * t_o \right), \tag{33}$$

where O^f represents the set of mathematical operations performed in function f, c_o^f is the number of times the mathematical operation o is executed in function f, t_o is the time required to execute the mathematical operation o, and t_f is the time required to execute DGKAP function f. If we use Equation (33) instead of Equation (16) for the rest of the analysis, we obtain results in terms of time costs of mathematical operations.

In our analysis, we assume that t_f or t_o have arbitrary but fixed values. However, we can also derive a more general result by expressing t_f or t_o in terms of protocol-specific parameters. For instance, let α and β be two protocol-specific parameters that affect t_f and t_o. In this case, t_f in Equation (16) and t_o in Equation (33) becomes $t_f(\alpha, \beta)$ and $t_o(\alpha, \beta)$, respectively. The terms with α and β are moved through derivations. We note that α and β are not random variables such as n or m. Instead, these are pre-defined parameters that are assigned constant values before the DGKAP operations start. Therefore, terms with α and β remain in the final expressions of the performance metrics W and L. By using different values for α and β parameters, we can estimate how the performance of DGKAP operation changes with respect to these parameters. In the rest of this sections, we apply our performance evaluation approach to different DGKAPs.

4.1 Illustration of the Performance Model on KAP-PBC [19]

In this subsection, we analyze the performance of join and mass join operations of Key Agreement Protocol with Partial Backward Confidentiality (KAP-PBC)[19]. The list of DGKAP functions that are used in KAP-PBC are as given in Table 6.

Based on the details given in Reference [19], P_n and P_{n-1} are the active participants in the group. It takes two rounds to complete a join operation with n participants in the group and m joining participants. In the first round, P_n executes Temporary Public Key Computation once and Temporary Public Key Broadcast for $n + m - 1$ participants. In the second round, each P_n and P_{n-1} executes Session Key Computation once, Session Key Broadcast for $n + m - 1$ participants, Session Key Verification for $m + 2$ participants, Temporary Public Key Verification for $m + 1$ participants, and Group Key Computation once, Therefore, for n participants in the group and m joining participants, we obtain

$$T_{max}^1 = t_{pkc} + (n + m - 1)t_{pkb}, \tag{34}$$

$$T_{max}^2 = t_{skc} + (n + m - 1)t_{skb} + (m + 2)t_{skv} + (m + 1)t_{pkv} + t_{gkc}. \tag{35}$$

Thus,

$$T_{max} = t_{pkc} + (n + m - 1)t_{pkb} + t_{skc} + (n + m - 1)t_{skb} + (m + 2)t_{skv} + (m + 1)t_{pkv} + t_{gkc}. \quad (36)$$

For single join operation, $m = 1$ and Equation (36) are reduced to

$$T_{max} = t_{pkc} + nt_{pkb} + t_{skc} + nt_{skb} + 3t_{skv} + 2t_{pkv} + t_{gkc}, \quad (37)$$

where T_{max} represents the mean service time of single join operation, $E[S_{sj}]$.

By replacing n terms in Equation (37) with the value of $E[n]$ in Equation (23), we obtain $E[S_{sj}]$ with constant terms as

$$E[S_{sj}] = 2t_{pkv} + 3t_{skv} + t_{skc} + t_{pkc} + t_{gkc} + (t_{skb} + t_{pkb})\left(\left\lfloor \frac{\lambda}{\mu} \right\rfloor + 0.5\right). \quad (38)$$

Then, we obtain the performance metrics of join operation as

$$W = \frac{1}{\frac{1}{E[S_{sj}]} - \lambda}, \quad (39)$$

$$L = \frac{\lambda}{\frac{1}{E[S_{sj}]} - \lambda}, \quad (40)$$

where $E[S_{sj}]$ is derived in Equation (38).

For the mass join operation, the mean service time is as given in Equation (36). We substitute L_q and $\left(\lfloor \frac{\lambda}{\mu} \rfloor + 0.5\right)$ into m and n, respectively, in Equation (36) to obtain

$$E[S_{mj}] = \left\lfloor \frac{\lambda}{\mu} \right\rfloor (t_{skb} + t_{pkb}) + L_q(t_{pkv} + t_{pkb} + t_{skb} + t_{skv}) + \tau_1, \quad (41)$$

where τ_1 is given by

$$\tau_1 = t_{pkv} + 2t_{skv} + t_{skc} + t_{pkc} + t_{gkc} - 0.5t_{skb} - 0.5t_{pkb}. \quad (42)$$

Then, we use Equation (28) and leave the L_q terms in one side to derive an upper bound for L_q as (see Appendix A.3 for intermediate steps)

$$W_q \leq \frac{\lambda(t_{skb} + t_{pkb}) + \mu\tau_1}{\mu - \lambda\mu(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}, \quad (43)$$

and we use $L_q = \lambda W_q$ to obtain an upper bound for L_q as

$$L_q \leq \frac{\lambda^2(t_{skb} + t_{pkb}) + \mu\tau_1}{\mu - \lambda\mu(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}. \quad (44)$$

In addition to single join and mass join operations, KAP-PBC also provides a third and novel operation for joining new participants, namely join with partial backward confidentiality (join-PBC). In this operation, a chosen joining participant first computes the group key just before the join operation that yields a single join operation in our performance model. Then, joining participants are added into the group with a mass join operation to compute the new group key. Therefore, we consider the join-PBC operation as a pipeline process of a single join and a mass join operation as shown in Figure 6.

For the analysis of join-PBC operation, the mean service time is calculated by using the following formula:

$$E[S_{pbc}] = E[S_{sj}] + E[S_{mj}], \quad (45)$$

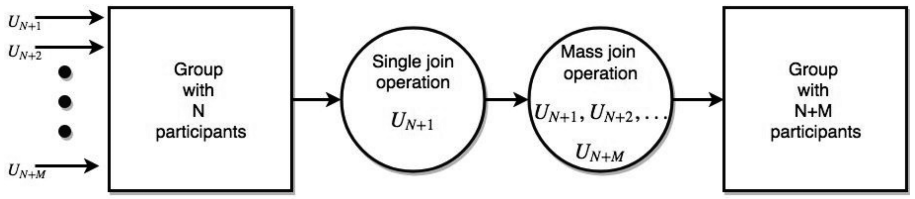

Fig. 6. An overview of join-PBC operation in KAP-PBC.

where S_{sj} is single join operation and S_{mj} is mass join operation. Therefore, we obtain a closed-from expression by summing the values of $E[S_{sj}]$ and $E[S_{mj}]$ in Equation (38) and Equation (41), respectively:

$$E[S_{pbc}] = 2\left\lfloor \frac{\lambda}{\mu} \right\rfloor (t_{skb} + t_{pkb}) + L_q(t_{pkv} + t_{pkb} + t_{skb} + t_{skv}) + \tau_1 + \tau_2, \tag{46}$$

where τ_2 is given as

$$\tau_2 = 2t_{pkv} + 3t_{skv} + t_{skc} + t_{pkc} + t_{gkc}. \tag{47}$$

Then, we obtain an upper bound for W_q and L_q (see Appendix A.5 for intermediate steps) as follows:

$$W_q \leq \frac{2\lfloor \frac{\lambda}{\mu} \rfloor (t_{skb} + t_{pkb}) + \tau_1 + \tau_2}{1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}, \tag{48}$$

$$L_q \leq \frac{2\lambda\lfloor \frac{\lambda}{\mu} \rfloor (t_{skb} + t_{pkb}) + \lambda\tau_1 + \lambda\tau_2}{1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}. \tag{49}$$

We can also include the effects of leave operation into single join and mass join operations. In KAP-PBC, when a participant P_i requests to leave, P_{i-1} and P_{i-2} are the active participants who perform the necessary operations. The probability that such participants decide to leave at the same time is low enough to ignore. Therefore, we assume that leave operations for different participants can be completed simultaneously, and we can use our leave model presented in Section 3.3.

In the first round of leave operation, P_{i-1} executes Temporary Public Key Computation and Temporary Public Key Broadcast functions. In the second round, P_{i-1} and P_{i-2} execute Temporary Public Key Verification, Secret Key Computation, Secret Key Distribution, Secret Key Verification, and Group Key Computation functions. Therefore, we obtain

$$T^1_{max} = t_{pkc} + (n-2)t_{pkb}, \tag{50}$$

$$T^2_{max} = t_{pkv} + t_{skc} + (n-2)t_{skd} + t_{skv} + t_{gkc}. \tag{51}$$

Thus,

$$E[S_l] = T_{max} = t_{pkc} + (n-2)t_{pkb} + t_{pkv} + t_{skc} + (n-2)t_{skd} + t_{skv} + t_{gkc}. \tag{52}$$

By replacing n terms with $E[n]$, we obtain

$$E[S_l] = t_{pkv} + t_{skv} + t_{pkc} + t_{skc} + t_{gkc} + \left(\left\lfloor \frac{\lambda}{\mu} \right\rfloor - 1.5\right)(t_{pkb} + t_{skb}). \tag{53}$$

The total residence time is the summation of $E[S_l]$ in Equation (53) and the mean residence time. The derivations are the same as we provide through Equations (72)–(76). Then, the total residence time of KAP-PBC is computed as

$$\mu_{total} = \frac{\mu}{\mu E[S_l] + 1}. \tag{54}$$

Table 7. DGKAP Functions Used in GKAP-MANET

DGKAP function	Parameter Symbol
Cluster Head Selection	t_{chs}
Temporary Public Key Distribution	t_{pkd}
Temporary Public Key Distribution	t_{pkd}
Temporary Public Key Verification	t_{pkv}
Secret Key Distribution	t_{skd}
Secret Key Verification	t_{skv}
Cluster Key Computation	t_{ckc}

Finally, we replace the μ terms in Equation (38) with μ_{total} in Equation (54) to derive the updated performance metrics of join, mass join, and join-PBC operations.

4.2 Illustration of Our Performance Model on GKAP-MANET

In this subsection, we analyze the performance of join and mass join operations of Group Key Agreement Protocol for Mobile Ad Hoc Networks (GKAP-MANET)[18]. The list of DGKAP functions that are used in GKAP-MANET is as given in Table 7.

For the analysis of join operation, we assume that there are n participants in the group and a new participant requests to join the group. Based on the details given in Reference [18], P_1, P_{n-1}, and P_n are the active participants in the group. It takes two rounds to complete the join operation. In the first round, P_n executes Temporary Public Key Computation and Temporary Public Key Distribution functions. In the second round, P_1, P_{n-1}, and P_n execute Temporary Public Key Verification, Secret Key Computation, Secret Key Distribution, Secret Key Verification, and Cluster Key Computation functions. Therefore, we obtain

$$T^1_{max} = t_{pkc} + (n-1)t_{pkd}, \tag{55}$$

$$T^2_{max} = t_{pkv} + t_{skc} + t_{skv} + (n-1)t_{skd} + t_{ckc}. \tag{56}$$

Thus,

$$E[S_{sj}] = T_{max} = t_{pkc} + (n-1)t_{pkd} + t_{pkv} + t_{skc} + t_{skv} + (n-1)t_{skd} + t_{ckc}. \tag{57}$$

By replacing n terms in Equation (57) with the value of $E[n]$ in Equation (23), we obtain $E[S_{sj}]$ with constant terms as

$$E[S_{sj}] = t_{pkc} + \left(\left\lfloor \frac{\lambda}{\mu} \right\rfloor - 0.5\right) t_{pkd} + t_{pkv} + t_{skc} + t_{skv} + \left(\left\lfloor \frac{\lambda}{\mu} \right\rfloor - 0.5\right) t_{skd} + t_{ckc}, \tag{58}$$

which corresponds to

$$E[S_{sj}] = t_{pkc} + t_{pkv} + t_{skc} + t_{skv} + t_{ckc} + \left(\left\lfloor \frac{\lambda}{\mu} \right\rfloor - 0.5\right)(t_{pkd} + t_{skd}). \tag{59}$$

Then, we obtain the performance metrics of join operation as

$$W = \frac{1}{\frac{1}{E[S_{sj}]} - \lambda}, \tag{60}$$

$$L = \frac{\lambda}{\frac{1}{E[S_{sj}]} - \lambda}, \tag{61}$$

where $E[S_{sj}]$ is derived in Equation (59).

For the mass join operation, we assume that the number of joining participants is m. Then, the time required to perform computations and communications in rounds 1 and 2 becomes

$$T^1_{max} = t_{pkc} + (n + m - 2)t_{pkd}, \tag{62}$$
$$T^2_{max} = mt_{pkv} + t_{skc} + (n + m - 2)t_{skd} + mt_{skv} + t_{ckc}. \tag{63}$$

Therefore, the mean service time of mass join operation, $E[S_{mj}]$, is calculated as

$$E[S_{mj}] = T_{max} = t_{pkc} + t_{skc} + (n + m - 2)t_{pkd} + (n + m - 2)t_{skd} + mt_{pkv} + mt_{skv} + t_{ckc}. \tag{64}$$

We substitute L_q and $\left(\lfloor\frac{\lambda}{\mu}\rfloor + 0.5\right)$ into m and n (in Equation (64)), respectively, to obtain the following equation:

$$E[S_{mj}] = T_{max} = t_{pkc} + t_{skc} + \left(\left\lfloor\frac{\lambda}{\mu}\right\rfloor + L_q - 1.5\right)t_{pkd} + \left(\left\lfloor\frac{\lambda}{\mu}\right\rfloor + L_q - 1.5\right)t_{skd} + L_q t_{pkv} + L_q t_{skv} + t_{ckc}. \tag{65}$$

Then, we obtain an upper bound for W_q and L_q (see Appendix A.4 for intermediate steps) as follows:

$$W_q \leq \frac{\lfloor\frac{\lambda}{\mu}\rfloor(t_{pkd} + t_{skd}) + t_{pkc} + t_{skc} + t_{ckc} - 1.5t_{pkd} - 1.5t_{skd}}{1 - \lambda(t_{pkd} + t_{skd} + t_{pkv} + t_{skv})}, \tag{66}$$

$$L_q \leq \frac{\lambda\lfloor\frac{\lambda}{\mu}\rfloor(t_{pkd} + t_{skd}) + \lambda t_{pkc} + \lambda t_{skc} + \lambda t_{ckc} - 1.5\lambda t_{pkd} - 1.5\lambda t_{skd}}{1 - \lambda(t_{pkd} + t_{skd} + t_{pkv} + t_{skv})}. \tag{67}$$

We can also include the effects of leave operation into single join and mass join operations. In GKAP-MANET, when a participant P_i requests to leave, P_{i+1}, P_{i-1}, and P_{i-2} are the active participants who perform the necessary operations. The probability that such participants decide to leave at the same time is low enough to ignore. Therefore, we assume that leave operations for different participants can be completed simultaneously, and we can use our leave model presented in Section 3.3.

In the first round of leave operation, P_{i+1}, P_{i-1}, and P_{i-2} execute Temporary Public Key Computation and Temporary Public Key Distribution functions. In the second round, P_{i+1}, P_{i-1}, and P_{i-2} execute Temporary Public Key Verification, Secret Key Computation, Secret Key Distribution, Secret Key Verification, and Cluster Key Computation functions. Therefore, we obtain

$$T^1_{max} = t_{pkc} + (n - 2)t_{pkd}, \tag{68}$$
$$T^2_{max} = t_{pkv} + t_{skc} + (n - 2)t_{skd} + t_{skv} + t_{ckc}. \tag{69}$$

Thus,

$$E[S_l] = T_{max} = t_{pkc} + (n - 2)t_{pkd} + t_{pkv} + t_{skc} + (n - 2)t_{skd} + t_{skv} + t_{ckc}. \tag{70}$$

By replacing n terms with $E[n]$, we obtain

$$E[S_l] = t_{pkc} + \left(\left\lfloor\frac{\lambda}{\mu}\right\rfloor - 1.5\right)t_{pkd} + t_{pkv} + t_{skc} + \left(\left\lfloor\frac{\lambda}{\mu}\right\rfloor - 1.5\right)t_{skd} + t_{skv} + t_{ckc}. \tag{71}$$

Table 8. Mathematical Operations Used in IBAAGKAP

Operation	Parameter Symbol
Scalar Exponentiation	t_{se}
Bilinear Map Computation	t_{bmc}
Hash Computation	t_{hc}
Update Message Table Row	t_{umtr}
Send Signature	t_{ss}

The total residence time is the summation of $E[S_l]$ in Equation (71) and the mean residence time:

$$E[S_{totalres}] = E[S_{res}] + E[S_l], \tag{72}$$

$$E[S_{totalres}] = \frac{1}{\mu} + E[S_l], \tag{73}$$

$$E[S_{totalres}] = \frac{\mu E[S_l] + 1}{\mu}. \tag{74}$$

Then, the total residence rate is computed as

$$\mu_{total} = \frac{1}{E[S_{totalres}]}, \tag{75}$$

$$\mu_{total} = \frac{\mu}{\mu E[S_l] + 1}. \tag{76}$$

Finally, we replace μ in Equation (59) as μ_{total} in Equation (76) to derive the updated performance metrics of join operation. Similarly, we replace μ in Equations (66) and (67) with μ_{total} to derive the updated upper bounds for performance metrics of mass join operations.

4.3 Illustration of the Performance Model on IBAAGKAP [48]

In this subsection, we analyze the performance of join operation of a recently proposed asymmetric group key agreement protocol namely Identity-Based Authenticated Asymmetric Group Key Agreement Protocol (IBAAGKAP)[48]. IBAAGKAP is an asymmetric protocol in which the encryption key is common and each participant has a unique decryption key. There is a group manager who maintains the group and actively plays a role in the operations.

Reference [48] does not define high-level functions such as Secret Key Verification in KAP-PBC. Therefore, we use the mathematical operations for the analysis. The list of mathematical operations used in IBAAGKAP are as given in Table 8.

For the analysis, we assume that there are n participants in the group and a new participant P_{n+1} requests to join the group. Based on the details given in Reference [48], the group manager and P_{n+1} are the active participants during the join operation. It takes one round to complete the operation. The group manager updates $n-1$ rows of the message table and sends signature to $n-1$ participants. P_{n+1} performs Scalar Exponentiation for $n+5$ times, Hash Computation for $n+1$ times, and Bilinear Map Computation for 4 times. Therefore, we obtain

$$T_{GM} = (n-1)(t_{umtr} + t_{ss}), \tag{77}$$

$$T_{P_{n+1}} = (n+5)t_{se} + (n+1)t_{bmc} + 4t_{hc}, \tag{78}$$

Table 9. Mathematical Operations Used in TGDH

Operation	Parameter Symbol
Modular Exponentiation	t_{me}
Sign Message	t_{sm}
Verify Message	t_{vm}
Send Tree Structure	t_{sts}

where T_{GM} and $T_{P_{n+1}}$ represents the time spent in join operation by the group manager and participant P_{n+1}, respectively. Then, the mean service time of join operation is derived as

$$E[S_j] = T_{max} = max\{(n-1)(t_{umtr} + t_{ss});\ (n+5)t_{se} + (n+1)t_{bmc} + 4t_{hc}\}. \tag{79}$$

By replacing n terms with $E[n]$, we obtain

$$E[S_j] = max\left\{\left(\left\lfloor\frac{\lambda}{\mu}\right\rfloor - 0.5\right)(t_{umtr} + t_{ss});\ \left(\left\lfloor\frac{\lambda}{\mu}\right\rfloor + 5.5\right)t_{se} + \left(\left\lfloor\frac{\lambda}{\mu}\right\rfloor + 1.5\right)t_{bmc} + 4t_{hc}\right\}. \tag{80}$$

Then, we obtain the performance metrics of join operation as

$$W = \frac{1}{\frac{1}{E[S_j]} - \lambda}, \tag{81}$$

$$L = \frac{\lambda}{\frac{1}{E[S_j]} - \lambda}. \tag{82}$$

4.4 Illustration of the Performance Model on TGDH [26]

In this subsection, we analyze the performance of join operation of a protocol called Tree-based Group Diffie Hellman (TGDH) [26]. The protocol utilizes key trees to achieve more efficient group key computation. The group is represented with a tree in which the session key of a participant is associated with a leaf.

We use mathematical operations for the analysis of join operation. The list of such operations used in TGDH are as given in Table 9.

For the analysis, we assume that there are n participants in the group and a new participant P_{n+1} requests to join the group. We also assume that the tree height is $h \geq 3$. Based on the details given in Reference [48], one of the participants takes a special role called sponsor who performs most of the computations and communications. In the first round, the sponsor performs 4 to $2h-2$ modular exponentiation operations, one message verification, one message sign, and sends the tree structure to n participants. In the second round, every participant performs 2 to $h-1$ modular exponentiation operations, one message sign, and two message verifications. Therefore, we obtain

$$4t_{me} + nt_{sts} + t_{sm} + t_{vm} \leq T^1_{max} \leq (2h-2)t_{me} + nt_{sts} + t_{sm} + t_{vm}, \tag{83}$$

$$2t_{me} + t_{sm} + 2t_{vm} \leq T^2_{max} \leq (h-1)t_{me} + t_{sm} + 2t_{vm}, \tag{84}$$

which produces the lower and upper bounds for T_{max} as

$$6t_{me} + nt_{sts} + 2t_{sm} + 3t_{vm} \leq T_{max} \leq (3h-3)t_{me} + nt_{sts} + 2t_{sm} + 3t_{vm}. \tag{85}$$

Thus, the mean service time of join operation is

$$6t_{me} + nt_{sts} + 2t_{sm} + 3t_{vm} \leq E[S_j] \leq (3h-3)t_{me} + nt_{sts} + 2t_{sm} + 3t_{vm}. \tag{86}$$

By replacing n terms with $E[n]$, we obtain

$$6t_{me} + \left\lfloor \frac{\lambda}{\mu} \right\rfloor t_{sts} + 2t_{sm} + 3t_{vm} \leq E[S_j] \leq (3h-3)t_{me} + \left\lfloor \frac{\lambda}{\mu} \right\rfloor t_{sts} + 2t_{sm} + 3t_{vm}. \tag{87}$$

Then, we derive bounds for our performance metrics as follows:

$$\frac{1}{\frac{1}{6t_{me}+\lfloor \frac{\lambda}{\mu} \rfloor t_{sts}+2t_{sm}+3t_{vm}} - \lambda} \leq W \leq \frac{1}{\frac{1}{(3h-3)t_{me}+\lfloor \frac{\lambda}{\mu} \rfloor t_{sts}+2t_{sm}+3t_{vm}} - \lambda}, \tag{88}$$

$$\frac{\lambda}{\frac{1}{6t_{me}+\lfloor \frac{\lambda}{\mu} \rfloor t_{sts}+2t_{sm}+3t_{vm}} - \lambda} \leq L \leq \frac{\lambda}{\frac{1}{(3h-3)t_{me}+\lfloor \frac{\lambda}{\mu} \rfloor t_{sts}+2t_{sm}+3t_{vm}} - \lambda}. \tag{89}$$

TGDH protocol tries to preserve the tree height, h, for join operations. The minimum value of h is $\lceil \log_2 n \rceil$. Since the protocol tries to preserve the height as much as possible, we can expect that

$$E[h] = \lceil \log_2 n \rceil, \tag{90}$$

$$E[h] \approx \left\lceil \log_2 \left\lfloor \frac{\lambda}{\mu} \right\rfloor \right\rceil. \tag{91}$$

By replacing h terms in Equations (88) and (89) with the approximated value of $E[h]$, we can further derive lower and upper bounds for our performance metrics as

$$\frac{1}{\frac{1}{6t_{me}+\lfloor \frac{\lambda}{\mu} \rfloor t_{sts}+2t_{sm}+3t_{vm}} - \lambda} \leq W \leq \frac{1}{\frac{1}{(3\lceil \log_2 \lfloor \frac{\lambda}{\mu} \rfloor \rceil-3)t_{me}+\lfloor \frac{\lambda}{\mu} \rfloor t_{sts}+2t_{sm}+3t_{vm}} - \lambda}, \tag{92}$$

$$\frac{\lambda}{\frac{1}{6t_{me}+\lfloor \frac{\lambda}{\mu} \rfloor t_{sts}+2t_{sm}+3t_{vm}} - \lambda} \leq L \leq \frac{\lambda}{\frac{1}{(3\lceil \log_2 \lfloor \frac{\lambda}{\mu} \rfloor \rceil-3)t_{me}+\lfloor \frac{\lambda}{\mu} \rfloor t_{sts}+2t_{sm}+3t_{vm}} - \lambda}. \tag{93}$$

4.5 Discussions on the Analytical Performance Models of Protocols

To give an insight for the comparison of the protocols, in this subsection, we discuss the analytical performance models derived for the protocols in References [18, 19, 26, 48]. As we expressed in the previous subsections, we use the average waiting time (W) and the average number of participants (L) as the performance metrics for the join operation. However, comparing the performance of these protocols by only using these metrics is not that straightforward, since each protocol has different mathematical operations and functions in the resulting formulations of these metrics. Therefore, we also consider the most time-consuming operation for DGKAPs, the number of executed modular exponentiation operations in each function as a baseline for the comparison as defined in Reference [19].

For KAP-PBC protocol, each function consists of only one modular exponentiation operation for t_{pkc}, t_{pkv}, t_{skc}, t_{skv}, t_{gkc}. Similarly, each of the following GKAP-MANET functions also consists of one modular exponentiation operation for t_{chs}, t_{pkv}, t_{skv}, t_{ckc}. When compared to the modular exponentiation in the resulting formulations of GKAP-MANET in Equation (59) and KAP-PBC in Equation (38), we expect that GKAP-MANET provides better performance than KAP-PBC. However, for the resulting formulation of IBAAGKAP protocol in Equation (80), there exist more modular exponentiation operations than KAP-PBC has in Equation (38). Therefore, we expect that KAP-PBC has better performance than IBAAGKAP. Finally, for the resulting formulation of TGDH in Equation (87), we can only obtain upper and lower bounds. Thus, it operates better in the best-case performance than KAP-PBC, and it operates worse than KAP-PBC in the worst-case performance.

Table 10. Simulated Execution Time for the Core Functions of KAP-PBC and GKAP-MANET Protocols

Function	Symbol		Execution Time (msec.)	
	KAP-PBC	GKAP-MANET	KAP-PBC	GKAP-MANET
Temporary Public Key Computation	t_{pkc}	t_{pkc}	611	536
Temporary Public Key Verification	t_{pkv}	t_{pkv}	15	6
Temporary Public Key Broadcast/Distribution	t_{pkb}	t_{pkd}	21	1
Session/Secret Key Computation	t_{skc}	t_{skc}	23	12
Session/Secret Key Verification	t_{skv}	t_{skv}	32	20
Session/Secret Key Broadcast	t_{skb}	t_{skd}	2	2
Group/Cluster Key Computation	t_{gkc}	t_{ckc}	1	10

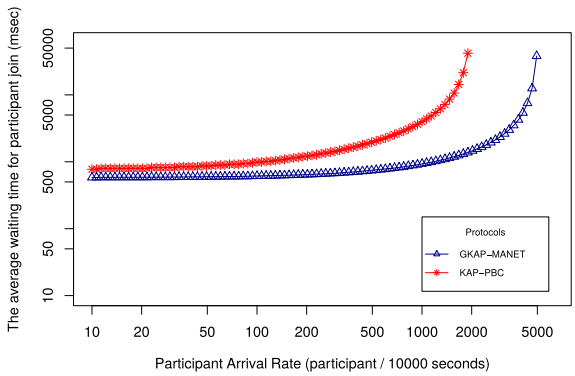

Fig. 7. The average waiting time of single join operation with respect to participant arrival rate for KAP-PBC and GKAP-MANET.

As a consequence, we observe that GKAP-MANET has the best performance and IBAAGKAP has the worst performance among other protocols.

To provide more precise comparison for the performance of DGKAPs, we also compare KAP-PBC [19] and GKAP-MANET [18]. Therefore, we use the simulation results for the execution time of each function as given in Table 10 by using simulation results on a computer with 1.8GHz Intel Core i5 processor and 4GB RAM. Figure 7 shows the average waiting time of single join operation for 15mins fixed residence time according to the given participant arrival rates. As shown in the figure, GKAP-MANET provides better performance than KAP-PBC as expected.

5 USING OUR PERFORMANCE MODEL ON SECURE FILE SHARING SYSTEMS

In this section, we extend our performance model to be used for secure file sharing systems (SFSS). First, we list our assumptions for the extension process. Then, we provide a demonstrative use case by choosing a SFSS and design a scenario where our performance model can be used to analyze

the performance of the chosen system. We reduce the number of possible scenarios by using the following assumptions:

- **Assumption 1** (Encryption/Decryption Time Complexity). *The time required to encrypt or decrypt a file depends only on the size of the file.*
- **Assumption 2** (Network Environment of Participants). *Participants in a group share the same characteristics in terms of networking capability. Participants have more or less the same Internet connection bandwidth.*
- **Assumption 3** (Constant Internet Connection Bandwidth). *The change in Internet connection bandwidth of participants is relatively small compared to the computational time complexity of the group key update operation.*

5.1 Demonstrative Example of Our Performance Model on an Example Secure File Sharing System

We employ Private Files Sharing System (PFSS) in Reference [19] as a demonstrative example. PFSS utilizes Key Agreement Protocol with Partial Backward Confidentiality (KAP-PBC). During the performance modelling of PFSS, we consider the service time as the combination of the file transmission time and the file encryption time.

In PFSS, files are stored in an encrypted manner. AES-256 in CBC mode is used to provide confidentiality. Assumption 1 holds, because AES has linear time complexity with respect to the size of encrypted files. In addition, with Assumption 2, the time required for a file transmission can be expressed for each participant as follows:

$$t_F = \frac{|\mathcal{F}|}{\mathcal{B}}, \tag{94}$$

where $|\mathcal{F}|$ is the size of the transmitted file and $\mathcal{B}$ is the network connection bandwidth. File transmission time, t_F, depends on the Internet connection bandwidth and the size of the transmitted file. Due to Assumption 3, $\mathcal{B}$ is considered as a constant value and t_F is linearly proportional to the size of the file. The transmission time has the same underlying probability distribution as the file size. Therefore, service time can be modelled by a file size distribution.

PFSS uses join-PBC function of KAP-PBC protocol to add new participants into the group. While executing the function, one of the new participants is selected to compute the group key just before joining the group. Therefore, this participant is able to re-encrypt shared group file(s) as shown in Figure 8. First, the selected participant joins the group by computing the group key just before joining the group. Then, all participants compute the new group key. After that, the selected participant re-encrypts the shared files by following steps shown in the Figure 8. The process is displayed in four steps: (i) downloads the shared group file(s), (ii) decrypts the file by using the existing group key, (iii) encrypts the file by using the new group key, (iv) uploads the re-encrypted file into the file system of PFSS.

To make more realistic numerical evaluations, we also consider studies in the literature regarding the distribution of file size such as in References [15, 21, 31, 32]. In general, the distribution of the size of a file can be expressed by using Pareto and Lognormal distributions. Therefore, we model the service time based on these distributions. Since Pareto and Lognormal distributions are non-Markovian, we assume that the service time is generally distributed. For simplicity, we also assume that the inter-arrival time is exponentially distributed. Therefore, we can use an $M/G/1$ queue for the file sharing operations as a general case. Instead, to deliver a more concrete example, we come up with a scenario and obtain numerical performance results based on this scenario.

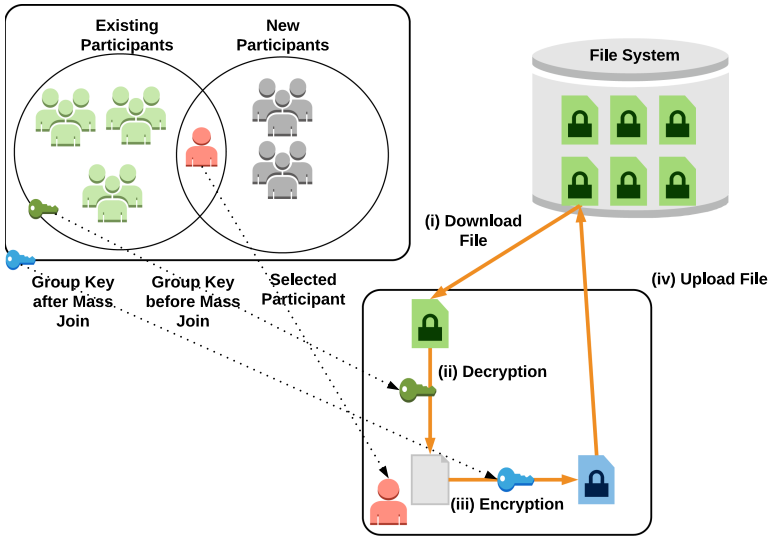

Fig. 8. An illustration of file re-encryption process in PFSS.

5.2 A Demonstrative Use-case Scenario for Healthcare System

We present an example use-case scenario for the use of Private File Sharing System [19] as a Healthcare System that securely stores medical records of patients. In this scenario, we assume that medical records of each patient are stored as an encrypted file in PFSS and each file is associated with a group of medical personnel. We also assume that an average size of files is approximately 300MB as reported in Reference [35]. Therefore, the chosen joining participant in our scenario re-encrypts a 300MB file after new participants join the group.

Performance metrics of PFSS in this scenario depend significantly on the Internet connection bandwidth. Consequently, we categorize the network connection bandwidth capacity into three different ranges for the analysis:

(1) **Less than 10Mbps:** In this case, the file download and the file upload operations are the bottleneck due to negligible group key update time. Since both the file size and the bandwidth are assumed as constant values, we use an $M/D/1$ queue to model upload and download operations. The expected service time is calculated as the time required to upload and download a medical file size, given by

$$E[S] = \frac{2 \times 300 \times 10^6}{\mathcal{B}} = \frac{6 \times 10^8}{\mathcal{B}} \ (sec) \ = \frac{6 \times 10^{11}}{\mathcal{B}} \ (msec), \tag{95}$$

where $\mathcal{B}$ is the Internet connection bandwidth in terms of bits per second (bps). As a result, the service rate is expressed as:

$$\mu = \frac{1}{E[S]} = \frac{\mathcal{B}}{6 \times 10^{11}}. \tag{96}$$

Note that performance metrics of an $M/G/1$ queue is derived in Pollaczek-Khinchine formula:

$$L \ = \frac{\rho^2 + \lambda^2 \sigma_s^2}{2(1-\rho)} + \rho, \tag{97}$$

$$W = \frac{\rho + \lambda\mu\sigma_s^2}{2(\mu - \lambda)} + \frac{1}{\mu}, \tag{98}$$

where ρ equals to λ/μ, λ is the participant arrival rate, μ is the service rate, and σ_s^2 is the variance of service time. By using the Pollaczek-Khinchine Formula with $\sigma_s^2 = 0$, we derive the mean number of waiting participants of $M/D/1$ queue as

$$L = \frac{\lambda^2}{2\mu(\mu - \lambda)} + \frac{\lambda}{\mu} \tag{99}$$

and the mean waiting time as

$$W = \frac{\lambda}{2\mu(\mu - \lambda)} + \frac{1}{\mu}. \tag{100}$$

(2) **Between 10Mbps and 1Gbps:** File operations and key update mechanism require approximately the same time. File operations are modelled by an $M/D/1$ queue, whereas the key update process, join-PBC operation, is modelled by an $M/M/1$ queue. The summation of one deterministic and one exponential service time is closer to exponential distribution. Therefore, we use a single server queue with exponential inter-arrival and service time to obtain a lower bound for performance metrics. The service time of the join-PBC operation is derived in Equation (46). We add the expected time required for the file download/upload and the file encryption/decryption operations into the expression in Equation (46) to obtain

$$\begin{aligned} E[S] = {} & 2\left\lfloor\frac{\lambda}{\mu}\right\rfloor(t_{skb} + t_{pkb}) + L_q(t_{pkv} + t_{pkb} + t_{skb} + t_{skv}) + \tau_1 + \tau_2 \\ & + t_{upl+dow} + t_{enc+dec}, \end{aligned} \tag{101}$$

where $t_{upl+dow}$ represents the total amount of time required to upload and download a file and $t_{enc+dec}$ represents the total amount of time required to encrypt and decrypt a file. For the average file size used in healthcare scenario, $t_{upl+dow}$ is

$$t_{upl+dow} = \frac{6 \times 10^{11}}{\mathcal{B}}\ (msec), \tag{102}$$

where $\mathcal{B}$ is the network connection bandwidth. Then, the mean service time is expressed by

$$\begin{aligned} E[S] = {} & 2\left\lfloor\frac{\lambda}{\mu}\right\rfloor(t_{skb} + t_{pkb}) + L_q(t_{pkv} + t_{pkb} + t_{skb} + t_{skv}) + \tau_1 + \tau_2 \\ & + \frac{6 \times 10^{11}}{\mathcal{B}} + t_{enc+dec}. \end{aligned} \tag{103}$$

Then, by combining the inequality in (26) and Equation (103), we derive an upper bound for the average waiting time of joining participants as given below:

$$W_q \leq \frac{2\lfloor\frac{\lambda}{\mu}\rfloor(t_{skb} + t_{pkb}) + \tau_1 + \tau_2 + \frac{6\times10^{11}}{\mathcal{B}} + t_{enc+dec}}{1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}, \tag{104}$$

where μ is the group residence rate, λ is the participant arrival rate of joining participants, and $\mathcal{B}$ is the Internet connection bandwidth. Also, we use $\tau 1$ and $\tau 2$ from Equation (42)

and (47), respectively. Then, by using $L_q = W_q\lambda$, we derive an upper bound for the average number of joining participants waiting in the queue:

$$L_q \leq \frac{2\lambda\lfloor\frac{\lambda}{\mu}\rfloor(t_{skb} + t_{pkb}) + \lambda\left(\tau_1 + \tau_2 + \frac{6\times10^{11}}{\mathcal{B}} + t_{enc+dec}\right)}{1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}. \tag{105}$$

(3) **More than 1Gbps:** File operation time is negligible compared to the time required to update the group key. We focus on the key update process and use a single server queue that is used to model join-PBC operation. The service time is considered as the time required for the join-PBC operation plus file encryption/decryption time. The service time is expressed as:

$$E[S] = 2\frac{\lambda}{\mu}(t_{skb} + t_{pkb}) + L_q(t_{pkv} + t_{pkb} + t_{skb} + t_{skv}) + \tau_1 + \tau_2 + t_{enc+dec}. \tag{106}$$

By making the same derivations in the previous case, we derive an upper bound for the performance metrics:

$$W_q \leq \frac{2\lambda(t_{skb} + t_{pkb}) + \mu(\tau_1 + \tau_2 + t_{enc+dec})}{\mu - \lambda\mu(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}, \tag{107}$$

$$L_q \leq \frac{2\lambda^2(t_{skb} + t_{pkb}) + \lambda\mu(\tau_1 + \tau_2 + t_{enc+dec})}{\mu - \lambda\mu(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}. \tag{108}$$

5.3 Numerical Results for the PFSS Use-case Scenario

In this subsection, we obtain numerical performance results for PFSS based on the healthcare scenario. We consider the average file size as 300MB. We also use the simulated execution time of KAP-PBC functions [19], which is as given in Table 10 in Section 4.5.

According to Ermis et al., the encryption/decryption time increases linearly with the size of the file in PFSS [19]. Moreover, it takes 3,772ms to encrypt/decrypt a 100MB file and 8,756ms to encrypt/decrypt a 500MB file on the platform specified in Table 10. Since the encryption/decryption time has linear relation to the size of a file, we can say that the encryption/decryption process would consume approximately $(3{,}772 + 8{,}756)/2 = 6{,}264$ms. Therefore, we conclude that $t_{enc+dec} = 2 * 6{,}264 = 12{,}528$ms. By using this approximation and execution times reported in Table 10, we derive the numerical results for performance metrics under three different bandwidth conditions:

(1) **The Internet connection bandwidth is less than 10Mbps:** In this case, the performance metrics are as follows:

$$W = \frac{\lambda}{2\mu(\mu - \lambda)} + \frac{1}{\mu}, \tag{109}$$

$$L = \frac{\lambda^2}{2\mu(\mu - \lambda)} + \frac{\lambda}{\mu}, \tag{110}$$

where λ is the participant arrival rate, $\mu = \frac{\mathcal{B}}{6\times10^{11}}$, and $\mathcal{B}$ is the Internet connection bandwidth.

(2) **The Internet connection bandwidth is between 10Mbps and 1Gbps:** In this case, the upper bounds for performance metrics are as follows:

$$W_q \leq \frac{2\lambda(2 + 21) + \mu(691 + 761 + \frac{6\times10^{11}}{\mathcal{B}} + 12528)}{\mu - \lambda\mu(15 + 21 + 2 + 32)}, \tag{111}$$

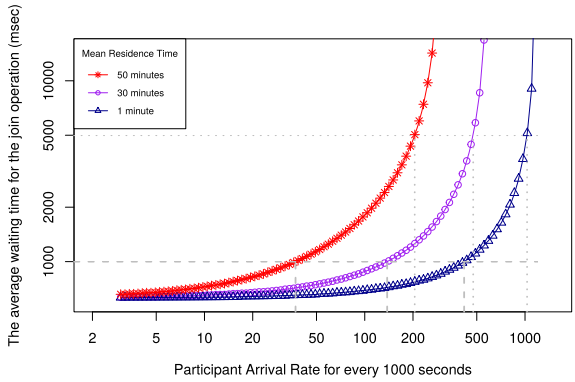

Fig. 9. Average waiting time in KAP-PBC against participant arrival rate for different mean residence time.

$$W_q \leq \frac{46\lambda + \mu(13980 + \frac{6\times 10^{11}}{\mathcal{B}})}{\mu - 70\lambda\mu}, \tag{112}$$

$$L_q \leq \frac{2\lambda^2(2+21) + \lambda\mu(691 + 761 + \frac{6\times 10^{11}}{\mathcal{B}} + 12528)}{\mu - \lambda\mu(15+21+2+32)}, \tag{113}$$

$$L_q \leq \frac{46\lambda^2 + \lambda\mu(13980 + \frac{6\times 10^{11}}{\mathcal{B}})}{\mu - 70\lambda\mu}. \tag{114}$$

(3) **The Internet connection bandwidth is more than 1Gbps:** In this case, upper bounds for performance metrics are as follows:

$$W_q \leq \frac{2\lambda(2+21) + \mu(691 + 761 + 12528)}{\mu - \lambda\mu(15+21+2+32)}, \tag{115}$$

$$W_q \leq \frac{46\lambda + 13980\mu}{\mu - 70\lambda\mu}, \tag{116}$$

(117)

$$L_q \leq \frac{2\lambda^2(2+21) + \lambda\mu(691 + 761 + 12528)}{\mu - \lambda\mu(15+21+2+32)}, \tag{118}$$

$$L_q \leq \frac{46\lambda^2 + 13980\lambda\mu}{\mu - 70\lambda\mu}. \tag{119}$$

The average waiting time of the join operation with respect to the participant arrival rate is shown in Figure 9 and Figure 10 for KAP-PBC and PFSS, respectively. We have analyzed the average waiting times for three different mean residence time values: 1min, 10mins, and 50mins. In our scenario, PFSS is used as a file sharing system, where medical personnel share records of patients among each other. We assume that each participant has a 1Gbps Internet connection bandwidth and the average waiting time for a participant before joining the group is between 20 and 60s. If we consider the waiting time threshold as 20s, then the limiting participant rates for each curve are 33 participants, 149 participants, and 664 participants at every 100s, respectively. If we consider the waiting time threshold of 60s, the limiting participant rates for each curve are 235 participants,

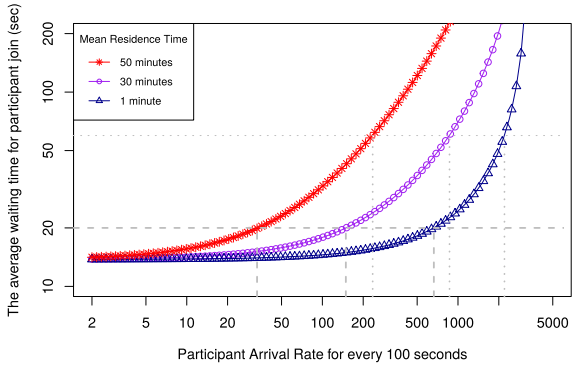

Fig. 10. Average waiting time of PFSS with respect to participant arrival rate for different mean residence time.

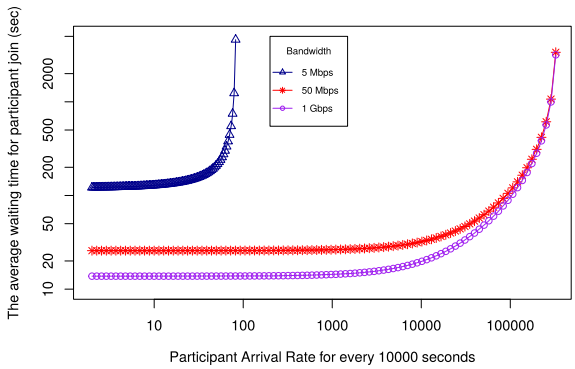

Fig. 11. Average waiting time of PFSS with respect to participant arrival rate for different Internet connection bandwidth.

866 participants, and 2,202 participants at every 100s, respectively. Therefore, we conclude that PFSS can be used in applications where the participant arrival rate is high and the mean residence time is less than a few minutes. In such applications, the average waiting time for joining a participant does not exceed 20ms as long as the participant arrival rate is less than 10 people per second, which is very high for a file sharing application. However, the average waiting time becomes significantly long if PFSS is used in an application with frequent arrival rate and long mean residence time.

We also present a graphical representation regarding the effect of participant bandwidth on the average waiting time in PFSS, as shown in Figure 11. We fix the mean residence time to 15mins and plot the average waiting time of the join operation with respect to the participant arrival rate for three different Internet connection bandwidth values: 5Mbps, 50Mbps, and 1Gbps. As shown in Figure 11, the average waiting time is significantly longer when the bandwidth is 5Mbps. In this case, file upload and download operations become the bottleneck. Even at a participant arrival rate of 100 participants at every 10,000s, the system becomes unavailable. Therefore, we deduce that PFSS should not be used in a file sharing application where the average file size is high but the Internet connection bandwidth of participants is low. When the bandwidth is 50Mbps or 1Gbps, results are similar. The difference between the average waiting time of the two cases is more visible when the participant arrival rate is low. When the participant arrival rate is higher, the number of

participants in the group increases and the file transmission time becomes negligible compared to the group key update time. Eventually, the average waiting time of the two cases converges when the participant arrival rate is approximately 10 participants per second.

6 CONCLUSION AND FUTURE WORK

In this article, we have proposed a general analytical performance model for the scalability of dynamic key agreement protocols. First, we have modelled the join and the mass join operations using networks of queues. Then, we have presented illustrations for the use of our model on different DGKAPS in References [18, 19, 26, 48]. In addition, we have derived the average waiting time for joining the group in terms of the participant arrival rate and the computational time complexity of cryptographic operations. Our performance models can help group key agreement protocol designers to estimate the average waiting time of joining participants with respect to the group residence time of participants. Moreover, we have proposed a model for the average waiting time of leaving participants.

To show the applicability of our model on a real-life scenario, we have extended the model for secure file sharing systems that use dynamic group key agreement protocols for the security of group communication. For a demonstrative example, PFSS in Reference [19] is employed as a health-care system scenario to securely share records of patients among medical personnel, since we have deduced that KAP-PBC could be used in applications with short residence time or low participant arrival rates such as one new participant at every 1,000s. We perform a numerical analysis on the scenario and conclude that PFSS should be used in applications where the participant arrival rate is high and the mean residence time is less than a few minutes. In addition, we obtain the average waiting time for joining participants with respect to the participant arrival rate for different bandwidth capacity values. We observe that using PFSS in applications where the average file is as large such as 500MB and the participants have low bandwidth capacity such as 5Mbps may violate the system availability. These results can be used for estimating the performance of a file sharing system. In addition, by adjusting the average file size and bandwidth parameters, our performance models can be applied to various applications. File sharing system admins can utilize these models to estimate the time required for the re-encryption process of a shared file.

We are planning to extend our analytical performance model as a framework to analyze the performance and scalability of dynamic group key agreement protocols to be used as a comparison tool for these protocols. Moreover, analysis of the applicability of the proposed models for different file sharing systems by employing well-known file size distributions and deriving performance metrics based on these distributions is another future research direction based on this study.

APPENDIX

A EQUATION DERIVATIONS

In this appendix, we present derivation details for equations in the article.

A.1 Derivations for Equation (7)–(8)

The performance metrics, W and L, of an $M/M/1$ queue are derived in Reference [37] as follows:

$$W = \frac{1}{\mu - \lambda}, \tag{120}$$

$$L = \frac{\lambda}{\mu - \lambda}, \tag{121}$$

where μ represents the service rate of the queue server and λ is the arrival rate. We can replace μ with $1/E[S]$. Thus,

$$W = \frac{1}{\frac{1}{E[S]} - \lambda}, \tag{122}$$

$$L = \frac{\lambda}{\frac{1}{E[S]} - \lambda}. \tag{123}$$

We replace $E[S]$ with the approximated value of $E[S_{join}]$ in Equation (6) to obtain

$$W = \frac{1}{\frac{1}{E[S_{join}]} - \lambda}, \tag{124}$$

$$L = \frac{\lambda}{\frac{1}{E[S_{join}]} - \lambda}. \tag{125}$$

A.2 Derivations for Equation (10)

The steady-state probability that there are k participants in an $M/G/\infty$ queue is given by the following expression [34]:

$$P_k = \frac{\lambda}{k\mu} P_{k-1}, \tag{126}$$

which reduces to

$$P_k = \left(\frac{\lambda}{\mu}\right)^k \frac{1}{k!} P_0. \tag{127}$$

Since $\sum_{k=0}^{\infty} P_k = 1$, we also derive P_0 in terms of λ and μ by using Taylor's Expansion:

$$P_0 \left(1 + \frac{\lambda}{\mu} + \frac{\left(\frac{\lambda}{\mu}\right)^2}{2!} + \frac{\left(\frac{\lambda}{\mu}\right)^3}{3!} + \cdots\right) = 1, \tag{128}$$

$$P_0 e^{\frac{\lambda}{\mu}} = 1, \tag{129}$$

$$P_0 = e^{-\frac{\lambda}{\mu}}. \tag{130}$$

By substituting P_0 in Equation (130) into Equation (127), we derive the closed-form expression for P_k as

$$P_k = \frac{\left(\frac{\lambda}{\mu}\right)^k e^{-\frac{\lambda}{\mu}}}{k!}. \tag{131}$$

Balance equations for the Mass Join Model are given as follows:

$$P_k \lambda = P_{k+1}(\lambda + \mu_{k+1}), \tag{132}$$

which equals to

$$P_k = \left(\frac{\lambda^k}{\prod_{i=1}^{k} (\lambda + \mu_i)}\right) P_0. \tag{133}$$

By substituting the above equation into $\sum_{k=0}^{\infty} P_k = 1$, we obtain

$$P_0 + P_0 \sum_{k=1}^{\infty} \left(\frac{\lambda^k}{\prod_{i=1}^{k} (\lambda + \mu_i)} \right) = 1, \tag{134}$$

$$P_0 = \left[1 + \sum_{k=1}^{\infty} \left(\frac{\lambda^k}{\prod_{i=1}^{k} (\lambda + \mu_i)} \right) \right]^{-1}, \tag{135}$$

as the closed-form expression of P_0.

A.3 Derivations for Equation (43)

The upper bound for the average waiting time of a mass join operation is derived as follows:

$$W_q \leq \frac{\lambda}{\mu}(t_{skb} + t_{pkb}) + L_q(t_{pkv} + t_{pkb} + t_{skb} + t_{skv}) + \tau_1, \tag{136}$$

$$W_q \leq \frac{\lambda}{\mu}(t_{skb} + t_{pkb}) + \lambda W_q(t_{pkv} + t_{pkb} + t_{skb} + t_{skv}) + \tau_1, \tag{137}$$

$$W_q(1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})) \leq \frac{\lambda}{\mu}(t_{skb} + t_{pkb}) + \tau_1, \tag{138}$$

$$W_q \leq \frac{\lambda(t_{skb} + t_{pkb}) + \mu\tau_1}{\mu - \lambda\mu(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}. \tag{139}$$

A.4 Derivations for Equations (66)–(67)

We substitute the expression in Equation (65) into Equation (26) to derive an upper bound for the average waiting time in the queue for joining participants as follows:

$$W_q \leq E[S_{mj}], \tag{140}$$

$$W_q \leq t_{pkc} + t_{skc} + \left(\left\lfloor \frac{\lambda}{\mu} \right\rfloor + L_q - 1.5 \right) t_{pkd} + \left(\left\lfloor \frac{\lambda}{\mu} \right\rfloor + L_q - 1.5 \right) t_{skd} + L_q t_{pkv} + L_q t_{skv} + t_{ckc}, \tag{141}$$

$$W_q \leq t_{pkc} + t_{skc} + \left(\left\lfloor \frac{\lambda}{\mu} \right\rfloor + \lambda W_q - 1.5 \right) t_{pkd} + \left(\left\lfloor \frac{\lambda}{\mu} \right\rfloor + \lambda W_q - 1.5 \right) t_{skd} + \lambda W_q t_{pkv} + \lambda W_q t_{skv} + t_{ckc}, \tag{142}$$

$$W_q \leq t_{pkc} + t_{skc} + t_{ckc} - 1.5t_{pkd} - 1.5t_{skd} + \left\lfloor \frac{\lambda}{\mu} \right\rfloor (t_{pkd} + t_{skd}) + \lambda W_q(t_{pkd} + t_{skd} + t_{pkv} + t_{skv}), \tag{143}$$

$$W_q - W_q\lambda(t_{pkd} + t_{skd} + t_{pkv} + t_{skv}) \leq t_{pkc} + t_{skc} + t_{ckc} - 1.5t_{pkd} - 1.5t_{skd} + \left\lfloor \frac{\lambda}{\mu} \right\rfloor (t_{pkd} + t_{skd}), \tag{144}$$

$$W_q \leq \frac{\lfloor \frac{\lambda}{\mu} \rfloor (t_{pkd} + t_{skd}) + t_{pkc} + t_{skc} + t_{ckc} - 1.5t_{pkd} - 1.5t_{skd}}{1 - \lambda(t_{pkd} + t_{skd} + t_{pkv} + t_{skv})}. \tag{145}$$

Then, by using Equation (145) and $L_q = \lambda W_q$, we compute an upper bound for L_q as follows:

$$\lambda W_q \leq \lambda \frac{\lfloor \frac{\lambda}{\mu} \rfloor (t_{pkd} + t_{skd}) + t_{pkc} + t_{skc} + t_{ckc} - 1.5t_{pkd} - 1.5t_{skd}}{1 - \lambda(t_{pkd} + t_{skd} + t_{pkv} + t_{skv})}, \tag{146}$$

$$L_q \leq \frac{\lambda \lfloor \frac{\lambda}{\mu} \rfloor (t_{pkd} + t_{skd}) + \lambda t_{pkc} + \lambda t_{skc} + \lambda t_{ckc} - 1.5\lambda t_{pkd} - 1.5\lambda t_{skd}}{1 - \lambda(t_{pkd} + t_{skd} + t_{pkv} + t_{skv})}. \tag{147}$$

A.5 Derivations for Equations (48)–(49)

We substitute the expression in Equation (46) into Equation (26) to derive an upper bound for the average waiting time of joining participants in join-PBC operation:

$$W_q \leq 2 \left\lfloor \frac{\lambda}{\mu} \right\rfloor (t_{skb} + t_{pkb}) + L_q(t_{pkv} + t_{pkb} + t_{skb} + t_{skv}) + \tau_1 + \tau_2, \tag{148}$$

$$W_q \leq 2 \left\lfloor \frac{\lambda}{\mu} \right\rfloor (t_{skb} + t_{pkb}) + \lambda W_q(t_{pkv} + t_{pkb} + t_{skb} + t_{skv}) + \tau_1 + \tau_2, \tag{149}$$

$$W_q(1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})) \leq 2 \left\lfloor \frac{\lambda}{\mu} \right\rfloor (t_{skb} + t_{pkb}) + \tau_1 + \tau_2, \tag{150}$$

$$W_q \leq \frac{2\lfloor \frac{\lambda}{\mu} \rfloor (t_{skb} + t_{pkb}) + \tau_1 + \tau_2}{1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}. \tag{151}$$

We also derive an upper bound for the average number of participants waiting to join by using Little's Law in Equation (151):

$$W_q \leq \frac{2\lfloor \frac{\lambda}{\mu} \rfloor (t_{skb} + t_{pkb}) + \tau_1 + \tau_2}{1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}, \tag{152}$$

$$\lambda W_q \leq \lambda \frac{2\lfloor \frac{\lambda}{\mu} \rfloor (t_{skb} + t_{pkb}) + \tau_1 + \tau_2}{1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}, \tag{153}$$

$$L_q \leq \frac{2\lambda \lfloor \frac{\lambda}{\mu} \rfloor (t_{skb} + t_{pkb}) + \lambda\tau_1 + \lambda\tau_2}{1 - \lambda(t_{pkv} + t_{pkb} + t_{skb} + t_{skv})}. \tag{154}$$

REFERENCES

[1] 1998. In *Proceedings of the Conference on Advances in Cryptology (EUROCRYPT'98)*. Vol. 1403. Helsinki, Finland, 127–144.

[2] Ahmed Abdel-Harfez, Ali Miri, and Luiz Orozco-Barbosa. 2007. Authenticated group key agreement protocols for ad hoc wireless networks. *Int. J. Netw. Secur.* 4 (2007), 90–98.

[3] Yair Amir, Yongdae Kim, Cristina Nita-Rotaru, and Gene Tsudik. 2004. On the performance of group key agreement protocols. *ACM Trans. Inf. Syst. Secur.* 7, 3 (Aug. 2004), 457–488.

[4] Daniel Augot, Raghav Bhaskar, Valerie Issarny, and Daniele Sacchetti. 2007. A three round authenticated group key agreement protocol for ad hoc networks. *Pervas. Mobile Comput.* 3 (2007), 36–52.

[5] M. Backes, C. Cachin, and A. Oprea. 2005. Lazy revocation in cryptographic file systems. In *Proceedings of the 3rd IEEE International Security in Storage Workshop (SISW'05)*.

[6] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. 2005. Hierarchical identity based encryption with constant size ciphertext. In *Proceedings of the Conference on Advances in Cryptology (EUROCRYPT'05)*, Ronald Cramer (Ed.). 440–456.

[7] Mike Burmester and Yvo Desmedt. 1994. A secure and efficient conference key distribution system (extended abstract). In *Proceedings of the Conference on Advances in Cryptology (EUROCRYPT'94)*. Springer, 275–286.

[8] Chin-Chen Chang, Hao-Chuan Tsai, Pen-Yi Chang Lijun Chu, Xuefeng Zheng, and Shaojie Wang. 2007. A collaborative conference key agreement scheme by using an intermediary node. In *Proceedings of the International Conference on Convergence Information Technology*. IEEE, 54–59.

[9] M. L. Chaudhry and J. G. C. Templeton. 1983. *A First Course in Bulk Queues*. Wiley, New York.

[10] Jiin-Chiou Cheng and Chi-Sung Laih. 2009. Conference key agreement protocol with non-interactive fault-tolerance over broadcast network. *Int. J. Inform. Sec.* 8 (2009), 1.

[11] Zi-Yao Cheng, Yun Liu, Chin-Chen Chang, and Cheng Guo. 2013. A fault-tolerant group key agreement protocol exploiting dynamic setting. *Int. J. Commun. Syst.* 26 (2013), 259–275.

[12] Cheong Hyeon Choi. 2013. Adaptation of Weil pairing IBE for secure file sharing. In *Proceedings of the 5th International Conference on Advances in Databases, Knowledge, and Data Applications (GlobeNet'13)*. 59–65.

[13] Yu-Fang Chung. 2013. The design of authentication key protocol in certificate-free public key cryptosystem. *Security and Communication Networks* 7, 11 (2013), 2125–2133. DOI: 10.1002/sec.924 (2013).

[14] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. 1992. Authentication and authenticated key exchanges. *Des. Codes Cryptogr.* 2 (1992), 107–125.

[15] Allen B. Downey. 2001. The structural cause of file size distributions. *SIGMETRICS Perform. Eval. Rev.* 29, 1 (June 2001), 328–329.

[16] Orhan Ermiş, Şerif Bahtiyar, Emin Anarım, and M. Ufuk Çağlayan. 2013. An improved fault-tolerant conference-key agreement protocol with forward secrecy. In *Proceedings of the 6th International Conference on Security of Information and Networks (SIN'13)*. 306–310.

[17] Orhan Ermiş, Şerif Bahtiyar, Emin Anarim, and M. Ufuk Çağlayan. 2017. A comparative study on the scalability of dynamic group key agreement protocols. In *Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES'17)*. 62:1–62:6.

[18] Orhan Ermiş, Şerif Bahtiyar, Emin Anarım, and M. Ufuk Çağlayan. 2017. A secure and efficient group key agreement approach for mobile ad hoc networks. *Ad Hoc Netw.* 67, C (Dec. 2017), 24–39.

[19] Orhan Ermis, Serif Bahtiyar, Emin Anarim, and M. Ufuk Caglayan. 2017. A key agreement protocol with partial backward confidentiality. *Comput. Netw.* 129, Part 1 (2017), 159–177.

[20] Rakesh Chandra Gangwar and Anil K. Sarje. 2006. Secure and efficient dynamic group key agreement protocol for an ad hoc network. In *Proceedings of the International Symposium on Ad Hoc and Ubiquitous Computing (ISAUHC'06)*. IEEE, 56–61.

[21] G. Gonçalves, I. Drago, A. P. C. d. Silva, A. B. Vieira, and J. M. Almeida. 2014. Modeling the dropbox client behavior. In *Proceedings of the IEEE International Conference on Communications (ICC'14)*. 1332–1337.

[22] K. H. Huang, E. C. Chang, and C. L. Chang. 2013. Secure file sharing scheme for mobile devices. In *Proceedings of the 4th International Conference on Networking and Distributed Computing*. 82–84.

[23] Kuo-Hsuan Huang, Yu-Fang Chung, Hsiu-Hui Lee, Feipei Lai, and Tzer-Shyong Chen. 2009. A conference key agreement protocol with fault-tolerant capability. *Comput. Stand. Interf.* 31 (2009), 401–405.

[24] Luan Ibraimi, Milan Petkovic, Svetla Nikova, Pieter Hartel, and Willem Jonker. 2009. Information security applications. In *Information Security Applications*, Heung Youl Youm and Moti Yung (Eds.). Springer-Verlag, Berlin, 309–323.

[25] Ingemar Ingemarsson, Donald T. Tang, and C. K. Wong. 1982. A conference key distribution system. *IEEE Trans. Inform. Theor.* 28 (1982), 714–719.

[26] Yongdae Kim, Adrian Perrig, and Gene Tsudik. 2004. Tree-based group key agreement. *ACM Trans. Inf. Syst. Secur.* 7, 1 (Feb. 2004), 60–96.

[27] Mingchu Li, Xiaodong Xu, Cheng Guo, and Xing Tan. [2016]. AD-ASGKA—Authenticated dynamic protocols for asymmetric group key agreement. *Secur. Commun. Netw.* 9, 11 (2016), 1340–1352.

[28] Qin Liu, Guojun Wang, and Jie Wu. 2014. Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. *Inf. Sci.* 258 (Feb. 2014), 355–370.

[29] Xuefeng Liu, Yuqing Zhang, Boyang Wang, and Jingbo Yan. 2013. Mona: Secure multi-owner data sharing for dynamic groups in the cloud. *IEEE Trans. Parallel Distrib. Syst.* 24, 6 (June 2013), 1182–1191.

[30] M. Malarvizhi, J. A. J. Sujana, and T. Revathi. 2014. Secure file sharing using cryptographic techniques in cloud. In *Proceedings of the International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE'14)*. IEEE, 1–6.

[31] Toshiko Matsumoto, Takashi Onoyama, and Norihisa Komoda. 2014. *Efficient Operational Management of Enterprise File Server with File Size Distribution Model*. Springer Netherlands, Dordrecht, 599–609.

[32] Michael Mitzenmacher. 2003. Dynamic models for file sizes and double Pareto distributions. *Internet Math.* 1, 3 (2003), 305–333.

[33] T. S. R. Murthy, Sivarama Krishna, and G. V. S Raju. 2012. Interdependent queueing model with varying bulk service. *Int. J. Math. Soft Comput.* 2, 1 (2012).

[34] G. F. Newell. 1966. The $M/G/\infty$ queue. *SIAM J. Appl. Math.* 14, 1 (1966), 86–88.

[35] Anthony J. Saibert. 2018. Archiving: Fundamentals of Storage Technology. Retrieved from: http://siim.org/page/archiving_chapter2.
[36] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. 2004. An analysis of live streaming workloads on the Internet. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC'04)*. 41–54.
[37] V. Sundarapandian. 2009. *Queueing Theory*. PHI Learning, India. 686–749 pages.
[38] Yang Tang, Patrick P. C. Lee, John C. S. Lui, and Radia Perlman. 2012. Secure overlay cloud storage with access control and assured deletion. *IEEE Trans. Depend. Secur. Comput.* 9, 6 (Nov. 2012), 903–916.
[39] J. Teng and C. Wu. 2016. An identity-based group key agreement protocol for low-power mobile devices. *Chinese J. Electron.* 25, 4 (2016), 726–733.
[40] Yuh-Min Tseng. 2005. An improved conference-key agreement protocol with forward secrecy. *Informatica, Lith. Acad. Sci.* 16 (2005), 275–284.
[41] Yuh-Min Tseng. 2007. A communication-efficient and fault-tolerant conference-key agreement protocol with forward secrecy. *J. Syst. Softw.* 80 (2007), 1091–1101.
[42] Wen-Guey Tzeng. 2000. A practical and secure fault-tolerant conference-key agreement protocol. In *Proceedings of the International Workshop on Public Key Cryptography*. Springer, 1–13.
[43] Wen-Guey Tzeng. 2002. A secure fault-tolerant conference-key agreement protocol. *IEEE Trans. Comput.* 51 (2002), 373–379.
[44] F. Wang, J. Liu, and Y. Xiong. 2008. Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *Proceedings of th 27th Conference on Computer Communications (IEEE INFOCOM'08)*. 1364–1372.
[45] Guojun Wang, Qin Liu, Jie Wu, and Minyi Guo. 2011. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Comput. Secur.* 30, 5 (July 2011), 320–331.
[46] Brent Waters. 2011. In *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography(PKC'11)*. Springer Berlin, 53–70.
[47] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. 2010. Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS'10)*. 261–270.
[48] L. Zhang, Q. Wu, J. Domingo-Ferrer, B. Qin, and Z. Dong. 2015. Round-efficient and sender-unrestricted dynamic group key agreement protocol for secure group communications. *IEEE Trans. Inform. Forens. Secur.* 10, 11 (2015), 2352–2364.
[49] Jianjie Zhao, Dawu Gu, and Yali Li. 2010. An efficient fault-tolerant group key agreement protocol. *Comput. Commun.* 33 (2010), 890–895.
[50] Zhongma Zhu and Rui Jiang. 2016. A secure anti-collusion data sharing scheme for dynamic groups in the cloud. *IEEE Trans. Parallel Distrib. Syst.* 27, 1 (Jan. 2016), 40–50.

Received August 2018; revised April 2019; accepted June 2019