On Generalized Divide and Conquer Approach for Group Key Distribution: Correctness and Complexity

***Abstract*—In this article we present a generalized version of divide and conquer approach for contributory group Diffie-Hellman key exchange (DHKE) scheme. In particular, we devise an efficient way to establish a mutual secret key for multiple participants that uses a quasilinear amount of exponentiations with respect to the number of participants. The correctness of our protocol is proven using mathematical induction. We also compute its complexity in terms of total exponentiations within the protocol, analyze several important computational characteristics, and analyze the security of the protocol against passive attack. Moreover, we provide a comprehensive comparison of our protocol with other existing contributory schemes. Finally, we present an adaptation of our protocol for Megrelishvili group key agreement as a variant of DHKE procedure.**

Keywords*—*divide and conquer approach*, *contributory group key distribution scheme

I. Introduction

The development of internet-based applications demands the improvement of network security. A secure communication in a network must support message confidentiality, data integrity, and user authenticity. A group key agreement protocol supports a construction of mutual secret key to communicate in an untrusted network [1]. There are several key management schemes: centralized, distributed, and contributory [2]. In a centralized scheme, there exists an entity that generates and distributes the mutual key over a trusted network; in a distributed scheme, each participant constructs the key independently. In a contributory scheme, each participant contributes in the generation of the mutual key. In this paper, we discuss a contributory scheme of group Diffie-Hellman Key Exchange (DHKE) protocol using divide and conquer (DC) strategy.

The DHKE protocol was first introduced by W. Diffie and M. Hellman in 1976 [3]. The research of the protocol for group key distribution has been widely conducted [4], [5], [6]. In [6], Steiner et al. proposed three schemes with the complexity of $\mathcal{O}(N^2)$ for GDH.1 and GDH.2 protocols, and $\mathcal{O}(N)$ for GDH.3 protocol. This complexity is measured based on the number of exponentiations in the entire protocol. In [7], Gaonkar and Pai stated that by using the divide and conquer strategy, the number of exponentiations performed by each participant can be reduced to no more than $\log_2 N + 1$ for a group of N participants. Consequently, the complexity of the protocol is $\mathcal{O}(N \log_2 N)$. Although the total number of exponentiations in DC protocol is greater than that in GDH.3, each participant performs the same amount of computations. This uniformity of computations does not happen in GDH.3, as there exists a participant that performs more computations than others. Nevertheless, the DC protocol in [7] is not explained in a general and formal manner. Therefore, the objectives of this article is to explicate the formal definition of the protocol and to analyze the correctness, complexity, and security of its algorithm. We generalize the protocol to allow an arbitrary number of participants ($N \geq 4$) in the protocol. Additionally, generalization of the protocol is performed in order to ease the analysis of the protocol mathematically. In addition, we also modify our DC strategy for Megrelishvili protocol as a variant of the DHKE.

The rest of this paper is organized as follows. Section II is reserved for terminologies and mathematical notations. In Section III, we discuss some existing group DHKE schemes and their characteristics. We present the generalization of DC DHKE protocol in Section IV along with its correctness, complexity, and security analysis. Section V discusses the adaptation of DC strategy for Megrelishvili protocol. Lastly, this paper is summarized and concluded in Section VI.

II. Notational Preliminaries

In this section, we first explain the notations and terminologies used in this paper. Some of the notations used are similar to those in [8]. We use $\mathbb{F}_p$ to denote the finite field of p elements and $\mathbb{F}_p^n$ to denote the n-dimensional vector space over $\mathbb{F}_p$. The primitive root or generator of the finite field is denoted by g. To denote vectors, we use boldface lowercase letters (e.g., $\mathbf{a}$) and to denote matrices, we use boldface uppercase letters (e.g., $\mathbf{B}$). All vectors are assumed as row vectors. The expression $\mathbf{aB}$ where $\mathbf{a} \in \mathbb{F}_p^n$ and $\mathbf{B}$ is a $n \times n$ matrix over $\mathbb{F}_p$ is called a right multiplication of $\mathbf{a}$ by $\mathbf{B}$.

Let p be an integer and a a positive integer. Then there are unique integers q and r with $0 \leq r \leq a - 1$ such that $p = aq + r$; p is called the divisor, a is called the dividend, q is called the quotient, and r is called the remainder. We use the operator rem to denote the remainder of a division, i.e., $r = p \operatorname{rem} a$ if and only if $p = aq + r$.

III. Existing Contributory Group Key Distribution Schemes

In this section, we explain some existing schemes for multi-party DHKE protocol. We consider a group of N participants, denoted by P_0, P_1, P_2, $\ldots$, P_{N-1}. The explanation for each protocol is as follows.

TABLE I. GDH.1 PROTOCOL.

Public Parameters Setup
All participants agree on a finite field $\mathbb{F}_p$ and its generator g.
Private Exponents Generation
For all $i \in [0, N-1]$, P_i chooses an integer α_i as its private exponent.
Upflow Phase
Upflow list initialization.
P_0 constructs a list $\mathsf{U} = [a^{\alpha_0}]$ and sends it to P_1.
For all $1 \leq i \leq N-2$.
1. P_i reads the last element of U and raises it to the power of α_i. 2. P_i appends the result to U. 3. P_i sends U to P_{i+1}.
Mutual key retrieval for P_{N-1}
P_{N-1} reads the last element of U and raises it to the power of α_{N-1}. The result is the mutual key.
Downflow Phase
Downflow list initialization.
$P_N - 1$ constructs a downflow list D where $\mathsf{D} = \left[a^{\left(\prod_{k=N-1}^{N-1} \alpha_k\right)\left(\prod_{k=0}^{j-1} \alpha_k\right)} : 0 \leq j \leq N-2 \right]$ and transmits it to P_{N-2}.
For each $1 \leq i \leq N-2$ in reverse order.
1. P_i takes the last element of D and raises it to the power of α_i. The result is the mutual key. 2. If $i \neq 0$, P_i raises all elements in D to the power of α_i. Afterward, P_i sends D to P_{i-1}.

A. Ingemarsson et al. Protocol (ING Protocol)

The ING protocol [5] uses a straightforward approach by arranging the participants in a circular configuration. At the beginning of the protocol, each participant P_i chooses a secret exponent α_i. The protocol uses $N-1$ phases of message transmission to establish the mutual key. Suppose $m_{i,j}$ denotes the message transmitted by participant P_i at the phase j where $0 \leq i, j \leq N-1$. In the initialization phase, each participant P_i computes $m_{i,0} = g^{\alpha_i}$. At the first phase, each P_i sends $m_{i,0}$ to $P_{(i+1) \text{ rem } N}$ and then computes $m_{i,1} = \left(m_{(i-1) \text{ rem } N,0}\right)^{\alpha_i}$. In general, at each phase j, where $1 \leq j \leq N-1$, participant P_i sends $m_{i,j-1}$ to $P_{(i+1) \text{ rem } N}$ and computes $m_{i,j} = \left(m_{(i-1) \text{ rem } N,j-1}\right)^{\alpha_i}$. At the end of the last phase (i.e., $(N-1)$-th phase), each participant P_i obtains the mutual key, i.e., $m_{i,(N-1)} = g^x$ where $x = \prod_{k=0}^{N-1} \alpha_k$. The total number of exponentiations in the ING protocol is N^2.

B. Burmester-Desmedt Protocol (BD Protocol)

The BD protocol in [4] uses three phases and each transmission consists only of message broadcasts. The transmission phases are as follows.

1) Each participant P_i chooses a secret number α_i and then broadcasts $z_i = g^{\alpha_i}$.
2) Each participant P_i calculates and broadcasts $X_i = (z_{i+1}/z_{i-1})^{\alpha_i}$.
3) P_i computes and obtains the mutual key $K = z_{i-1}^{N\alpha_i} \cdot X_i^{N-1} \cdot X_{i+1}^{N-2} \ldots X_{i-2} \mod p$.

The mutual key generated by this protocol is not the regular DH key, namely $K = g^{\alpha_0\alpha_1+\alpha_1\alpha_2+\cdots+\alpha_{N-1}\alpha_0}$. The total number of exponentiations in BD protocol is $N(N+1)$.

C. GDH.1, GDH.2, and GDH.3

GDH.1, GDH.2, and GDH.3 was introduced by Steiner et al. in [6]. The GDH.1 needs two phases in generating the mutual key: upflow and downflow. The upflow phase is used to collect contributions from each participant, whereas the downflow phase distributes the key for each of the participants. The total number of exponentiations in GDH.1 is $((N+3)N)/2 - 1$. We summarize the GDH.1 protocol in Table I.

The GDH.2 needs two phases in generating the mutual key: upflow and broadcast. Unlike the GDH.1 scheme, GDH.2 scheme allows all but one participants to retrieve the mutual key simultaneously. Similar with GDH.1 protocol, the total number of exponentiations in GDH.2 is $((N+3)N)/2 - 1$. We outline the GDH.2 protocol in Table II.

In contrast to the two previous schemes, the GDH.3 needs four phases in generating the mutual key: upflow, response, and two broadcast phases. The total number of exponentiations in GDH.3 is $5N - 6$. Consequently, this protocol is more efficient than the previous two schemes in [6]. We summarize the GDH.3 protocol in Table III.

IV. DIVIDE AND CONQUER METHOD FOR GROUP KEY ESTABLISHMENT

In this section we describe the generalized DC method for DHKE key agreement, as well as its correctness, complexity, and security analysis. The correctness is determined by whether or not all participants obtain the same mutual key at the end of the protocol. The complexity is measured based on the number of exponentiations in the entire protocol. While the security is shown only against passive attacker.

A. Protocol Description

The divide and conquer approach for multi-party DHKE was first proposed by Gaonkar and Pai in [7]. However, the original algorithm only provides the protocol explanation for a group of eight participants. There is no rigorous and general description about the protocol in a formal manner. In this

TABLE II. GDH.2 PROTOCOL.

Public Parameters Setup
The public parameters are identical to those described in Table I.
Private Exponents Generation
The private exponents generation is identical to those described in Table I.
Upflow Phase
Upflow list initialization.
P_0 constructs the list $\mathsf{U} = [a^{\alpha_0}]$ and sends it to P_1. P_1 updates the list $\mathsf{U} = [a^{\alpha_0\alpha_1}, a^{\alpha_0}, a^{\alpha_1}]$ and transmits it to P_2.
For all $2 \leq i \leq N-2$.
1. P_i receives the list U from P_{i-1} 2. P_i constructs the list U' using the rules $\mathsf{U}'[1] = \mathsf{U}[0]$, $\mathsf{U}'[0] = \mathsf{U}[0]^{\alpha_i}$, and $\mathsf{U}'[j] = \mathsf{U}[j-1]$ for each $2 \leq j \leq i+1$. 3. P_i sends U' to P_{i+1}.
Mutual key retrieval for P_{N-1}
P_{N-1} reads the first element of list U and raises it to the power of α_{N-1}. The result is the mutual key.
Broadcast Phase
Broadcast list initialization.
$P_N - 1$ constructs the broadcast list where $\mathsf{B}[j] = \mathsf{U}[j+1]^{\alpha_{N-1}}$ for each $0 \leq j \leq N-2$ and broadcasts B to all P_i where $0 \leq i \leq N-2$.
Mutual key retrieval for P_i where $0 \leq i \leq N-2$
P_i calculates $\mathsf{B}[N-2-i]^{\alpha_i}$ to obtain the mutual key.

TABLE III. GDH.3 PROTOCOL.

Public Parameters Setup
The public parameters are identical to those described in Table I.
Private Exponents Generation
The private exponents generation is identical to those described in Table I.
Upflow Phase
Upflow value initialization.
P_0 computes a^{α_0} and sends it to P_1.
For all $1 \leq i \leq N-2$
1. P_i receives $a^{\prod_{k=0}^{i-1}\alpha_k}$ from P_{i-1}. 2. P_i calculates $\left(a^{\prod_{k=0}^{i-1}\alpha_k}\right)^{\alpha_i}$ and transmits the value to P_{i+1}.
Mutual key retrieval for P_{N-1}
P_{N-1} raises the value received from P_{N-2} to the power of α_{N-1}. In other words, P_{N-1} computes $\left(a^{\prod_{k=0}^{N-2}\alpha_k}\right)^{\alpha_{N-1}} = a^{\prod_{k=0}^{N-1}\alpha_k}$ as the mutual key.
First Broadcast Phase
P_{N-2} transmits the value of $a^{\prod_{k=0}^{N-2}\alpha_k}$ to all P_i where $0 \leq i \leq N-3$.
Response Phase
For all $0 \leq i \leq N-2$
1. P_i calculates α_i^{-1}, i.e., the inverse of its private exponent. 2. P_i computes $\left(a^{\prod_{k=0}^{N-2}\alpha_k}\right)^{\alpha_i^{-1}} = a^{\prod_{k=0\wedge k\neq i}^{N-2}\alpha_k}$. 3. P_i defines $\mathsf{S}[i] = a^{\prod_{k=0\wedge k\neq i}^{N-2}\alpha_k}$ and sends $\mathsf{S}[i]$ to P_{N-1}.
Second Broadcast Phase
1. P_{N-1} constructs the broadcast list B where $\mathsf{B}[j] = \mathsf{S}[j]^{\alpha_{N-1}}$ for all $0 \leq j \leq N-2$. 2. P_{N-1} sends B to all P_i where $0 \leq i \leq N-2$.
Mutual key retrieval for P_i where $0 \leq i \leq N-2$
To obtain the mutual key, P_i reads $\mathsf{B}[i]$ and computes $\mathsf{B}[i]^{\alpha_i}$.

section, we discuss the formal definition of the protocol. We first consider a group of N participants where $N = 2^k$ and $k \in \mathbb{Z}$, denoted by P_0, P_1, ..., P_{N-1}. At the beginning, all participants agree on several public parameters, i.e.: a finite field $\mathbb{F}_p$ and a generator g in $\mathbb{F}_p$. Afterwards, each participant P_i chooses α_i as its secret exponent.

In the first transmission phase, all participants are divided into two groups, each group consists of $N/2$ participants. Initially, P_0 computes g^{α_0} and transmits it to P_1. Simultaneously, $P_{N/2}$ also calculates $g^{\alpha_{N/2}}$ and sends it to $P_{(N/2+1) \operatorname{rem} N}$. Then, for all $1 \leq i \leq N/2-1$ and $N/2+1 \leq i \leq N-1$, participant P_i raises the value received from P_{i-1} to the power of its secret exponent (α_i), and sends the result to P_{i+1}. With the exception of participant $P_{N/2-1}$ who transmits the result to $P_{N/2}$ and $P_{3N/4}$, and participant P_{N-1} who sends the result to P_0 and $P_{N/4}$.

For each transmission phase r where $2 \leq r \leq \log_2 N$, each group from the previous phase performs identical processes simultaneously. Thus, in each phase r, there are 2^{r-1} parallel processes. Finally, in the $(\log_2(N)+1)$-th phase, the number

TABLE IV. DIVIDE AND CONQUER APPROACH FOR MULTI-PARTY DHKE.

Public Parameter Setup
The public parameters are identical to those described in Table I
Private Exponents Generation
The private exponents generation is identical to those described in Table I.
Number of Participants Checking
if $N \neq 2^k$ where $k \in \mathbb{Z}$ then Add $2^{\lceil \log_2 N \rceil} - N$ dummies among the existing participants.
Phase 1
Initialization
All participants are divided into two groups, each group consists of $N/2$ participants. P_0 calculates g^{α_0} and sends it to P_1. $P_{N/2}$ also computes $g^{\alpha_{N/2}}$ and sends it to $P_{(N/2+1) \text{ rem } N}$.
For all $1 \leq i \leq N/2 - 1$ and $N/2 + 1 \leq i \leq N - 1$.
P_i raises the value received from P_{i-1} to the power of α_i. If $i = N/2 - 1$ or $i = N - 1$: $P_{N/2-1}$ transmits the result to $P_{N/2}$ and $P_{3N/4}$. P_{N-1} sends the result to P_0 and $P_{N/4}$. Else: P_i transmits the result to P_{i+1}.
r-th Phase, where $2 \leq r \leq \log_2 N$
Each group from the previous phase performs the same process simultaneously. Thus, in each phase r, there are 2^{r-1} parallel processes. The base of the exponentiation is the values received from the $(r-1)$-th phase.
$(\log_2 N + 1)$-th Phase
The number of participants in each process is equal to one. To retrieve the mutual key, each participant P_i raises the value received from the previous phase to the power of α_i.

of participants in each process is equal to one. To get the mutual key, each participant P_i raises the value received from the previous phase to the power of α_i. Note that the protocol needs $\log_2(N) + 1$ phases for all participants to receive the mutual key, i.e., g^x where $x = \prod_{k=0}^{N-1} \alpha_k$.

If the number of participants is not a power of two ($N \neq 2^k$ where $k \in \mathbb{Z}$), the DC protocol cannot be used directly. The reason is, in each transition between phases, the number of participants is halved. To overcome this issue, we add some dummy participants so that the initial number of participants in the group is equal to $2^{\lceil \log_2 N \rceil}$. This process, called padding, is performed by adding $2^{\lceil \log_2 N \rceil} - N$ dummies among the existing participants. The computations of these dummies are performed by the participant in the previous sequence. To avoid burdening other participants' computations, the secret exponents of these dummy participants are assigned to one. For example, consider a group of $N = 5$ participants. We add three dummy participants with the following configuration: P_1, P_6, P_2, P_7, P_3, P_4, where P_5, P_6, and P_7 are dummies.

Suppose the list E is a list of the secret exponents of all participants, and m_i is the value computed by participant P_i. We summarize the protocol in Table IV and present its simulation procedure in Algorithm 1. We give the example of the DC approach for four participants in Example 1.

Example 1 For $N = 4$, each of participants P_0, P_1, P_2, and P_3 chooses the secret exponent α_0, α_1, α_2, and α_3, respectively. Based of Algorithm 1, the construction of mutual key consists of three phases as follows:

Phase 1

- P_0 calculates g^{α_0} and sends it to P_1.
- Simultaneously, P_2 calculates g^{α_2} and sends it to P_3.
- P_1 calculates $g^{\alpha_0\alpha_1}$ and sends it to P_2 and P_3.
- P_3 also computes $g^{\alpha_2\alpha_3}$ and sends it to P_0 and P_1.

Phase 2

- P_0 calculates $g^{\alpha_2\alpha_3\alpha_0}$ and sends it to P_1.
- P_1 computes $g^{\alpha_2\alpha_3\alpha_1}$ and transmits it to P_0.
- P_2 calculates $g^{\alpha_0\alpha_1\alpha_2}$ and sends it to P_3.
- P_3 computes $g^{\alpha_0\alpha_1\alpha_3}$ and sends it to P_2.

Phase 3

- P_0 calculates $g^{\alpha_2\alpha_3\alpha_1\alpha_0}$ as the mutual key.
- P_1 computes $g^{\alpha_2\alpha_3\alpha_0\alpha_1}$ as the mutual key.
- P_2 calculates $g^{\alpha_0\alpha_1\alpha_3\alpha_2}$ as the mutual key.
- P_3 computes $g^{\alpha_0\alpha_1\alpha_2\alpha_3}$ as the mutual key.

The process described in Example 1 can be expounded using message sequence chart in Fig. 1.

B. Correctness Analysis

To simplify the correctness analysis of the protocol, we consider only the exponent values received by each participant. Those values are stored in an array K_r. The formal definition of the array K_r is as follows.

Definition 1 K_r is an array of exponents received by all participants after the r-th phase. The index i in K_r denotes the exponent value received by the i-th participant.

Theorem 1 *At the end of the protocol with $N = 2^k$ where $k \in \mathbb{Z}$ and $N \geq 8$, each participant receives the mutual key, namely g^x, where $x = \prod_{i=0}^{2^k-1} \alpha_i$ after $k + 1$ transmission phases.*

Proof: We prove Theorem 1 by mathematical induction on N. For $N = 4$, the theorem is correct according to Example 1.

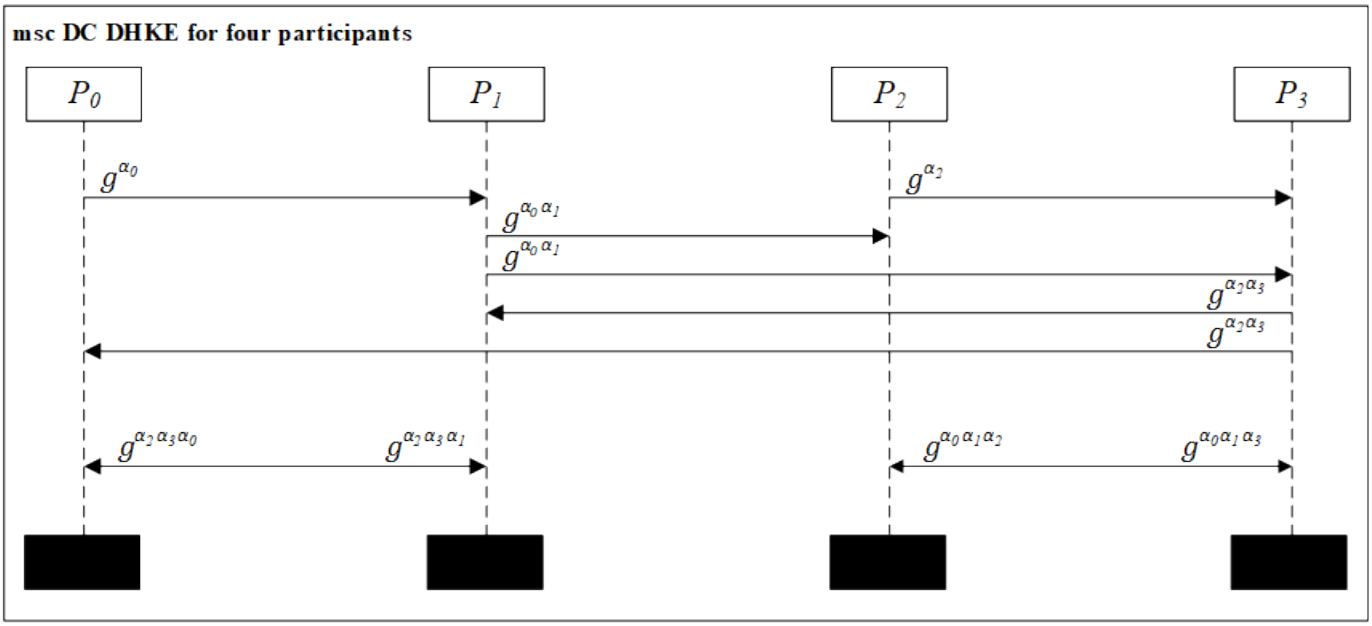

Fig. 1. Message Sequence Chart for DC DHKE of four participants.

For $N > 4$, we assume Theorem 1 is correct for $N = 2^k$, then we will prove that Theorem 1 holds for $N = 2^{k+1}$. Induction is performed from $N = 2^k$ to $N = 2^{k+1}$ because we assume that the number of participants is a power of two. We will prove that at the end of key exchange with $N = 2^{k+1}$, all participants acquire the mutual key g^x where $x = \prod_{i=0}^{2^{k+1}-1} \alpha_i$ in $k+2$ transmission phases. Observe that at the end of the first phase where $N = 2^{k+1}$, the array K_1 is as follows:

$$\mathsf{K}_1 = \left[\prod_{i=0}^{0} \alpha_i, \ldots, \prod_{i=0}^{2^k-1} \alpha_i, \prod_{i=2^k}^{2^k} \alpha_i, \ldots, \prod_{i=2^k}^{2^{k+1}-1} \alpha_i \right].$$

We denote $A = \prod_{i=0}^{2^k-1} \alpha_i$ and $B = \prod_{i=2^k}^{2^{k+1}-1} \alpha_i$. Then, at the end of second phase, the array K_2 is:

$$\mathsf{K}_2 = \left[B \prod_{i=0}^{0} \alpha_i, \ldots, B \prod_{i=0}^{2^{k-1}-1} \alpha_i, B \prod_{i=2^{k-1}}^{2^{k-1}} \alpha_i, \ldots, B \prod_{i=2^{k-1}}^{2^k-1} \alpha_i, \right.$$
$$\left. A \prod_{i=2^k}^{2^k} \alpha_i, \ldots, A \prod_{i=2^k}^{3(2^{k-1})-1} \alpha_i, A \prod_{i=3(2^{k-1})}^{3(2^{k-1})} \alpha_i, \ldots, A \prod_{i=3(2^{k-1})}^{2^{k+1}-1} \alpha_i \right].$$

At the second phase, there are two simultaneous processes with each number of participants are $N = 2^k$. At the first process, the participants are P_0, …, P_{2^k-1}. According to the induction hypothesis, at the end of key exchange with $N = 2^k$, each participant obtains the exponent value $\prod_{i=0}^{2^k-1} \alpha_i$. However, according to K_2, each one of those values is multiplied by B, a value transmitted by $P_{2^{k+1}-1}$ at the end of the first phase. Consequently, at the end of the first process, each participant obtains the exponent value $B \prod_{i=0}^{2^k-1} \alpha_i = \left(\prod_{i=2^k}^{2^{k+1}-1} \alpha_i \right) \left(\prod_{i=0}^{2^k-1} \alpha_i \right) = \prod_{i=0}^{2^{k+1}-1} \alpha_i$.

At the second process, the participants are $P_{2^k}, \ldots, P_{2^{k+1}-1}$. According to the induction hypothesis, at the end of key exchange, each participant obtains the exponent value $\prod_{i=2^k}^{2^{k+1}-1} \alpha_i$. However, according to K_2, each one of those values is multiplied by A, a value transmitted by P_{2^k-1} at the end of the first phase. Consequently, at the end of the second process, each participant obtains the exponent value $A \prod_{i=2^k}^{2^{k+1}-1} \alpha_i = \left(\prod_{i=0}^{2^k-1} \alpha_i \right) \left(\prod_{i=2^k}^{2^{k+1}-1} \alpha_i \right) = \prod_{i=0}^{2^{k+1}-1} \alpha_i$.

Both processes are performed simultaneously. Therefore, according to the induction hypothesis, after the first phase, there are $k+1$ other transmission phases until the key exchange is completed. Consequently, for $N = 2^{k+1}$, there are $k+1+1 = k+2$ transmission phases for all participants to obtain the mutual key. □

C. Complexity Analysis and Computational Characteristics

We use Lemma 1 to ease our complexity analysis of the protocol.

Lemma 1 *For $N = 2^k$ where $k \in \mathbb{Z}$ and $N \geq 4$, in each transmission phase of Algorithm 1, each participant only performs one exponentiation.*

Proof: According to line 5, 6, 10, and 16 of Algorithm 1, it is obvious that each P_i, where $0 \leq i \leq N-1$, performs only one exponentiation. □

We consider a group of N participants. After padding, the number of participants is $M = 2^{\lceil \log_2 N \rceil}$, which is less than or equal to twice of the original group size. We will prove the following Corollary 1.

Algorithm 1 Divide and Conquer Approach for multi-party DHKE Algorithm.

procedure DIVIDECONQUER(N, a, $\mathsf{E}[0 \dots N-1]$)
  if $N = 1$ **then**
    P_0 computes $a^{\mathsf{E}[0]}$
  else
    P_0 calculates $m_0 = a^{\mathsf{E}[0]}$
    $P_{N/2}$ computes $m_{N/2} = a^{\mathsf{E}[N/2]}$
    P_0 sends m_0 to P_1
    $P_{N/2}$ transmits $m_{N/2}$ to $P_{(N/2+1) \operatorname{rem} N}$
    for $i \leftarrow 1$ **to** $N/2-1$
      P_i computes $m_i \leftarrow (m_{i-1})^{\mathsf{E}[i]}$
      if $i \neq N/2-1$ **then**
        P_i sends m_i to P_{i+1}
      else
        $P_{N/2-1}$ transmits $m_{N/2-1}$ to $P_{N/2}$ and $P_{3N/4}$
    for $i \leftarrow N/2+1$ **to** $N-1$
      P_i calculates $m_i \leftarrow (m_{i-1})^{\mathsf{E}[i]}$
      if $i \neq N-1$ **then**
        P_i sends m_i to P_{i+1}
      else
        P_{N-1} transmits m_{N-1} to P_0 and $P_{N/4}$
    DIVIDECONQUER($N/2$,m_{N-1},$\mathsf{E}[0 \dots N/2-1]$)
    DIVIDECONQUER($N/2$,$m_{N/2-1}$,$\mathsf{E}[N/2 \dots N-1]$)
  end if
end procedure
if ($N \geq 8$) and ($N = 2^k$ for $k \in \mathbb{Z}$) **then**
  DIVIDECONQUER(N,a,$\mathsf{E}[0 \dots N-1]$)
else if ($N \geq 8$) and ($N \neq 2^k$ for $k \in \mathbb{Z}$) **then**
  $idx \leftarrow 0$
  for $i \leftarrow N - (2^{\lceil log_2 N \rceil} - N)$ **to** $(N-1)$ ▷ Padding
    Insert P_i between P_{idx} and P_{idx+1}.
    Insert 1 between $\mathsf{E}[idx]$ and $\mathsf{E}[idx+1]$.
    $idx \leftarrow idx + 1$
  Adjust the order of the participants
  DIVIDECONQUER(N,a,$\mathsf{E}[0 \dots 2^{\lceil \log_2 N \rceil}]$)
end if

Corollary 1 *The total amount of exponentiations in the divide and conquer DHKE protocol is* $2^{\lceil \log_2 N \rceil}(\log_2 \lceil \log_2 N \rceil + 1) = \mathcal{O}(N \log_2 N)$.

Proof: According to Theorem 1, there are $\log_2 M + 1$ phases for M participants. Lemma 1 states that each participant performs one exponentiation in each phase. Consequently, the total number of exponentiations in the protocol is $M(\log_2 M + 1)$. We have $M = 2^{\lceil \log_2 N \rceil}$, therefore $M(\log_2 M + 1) = 2^{\lceil \log_2 N \rceil}(\lceil \log_2 N \rceil + 1) = \mathcal{O}(N \log_2 N)$. □

We conduct a numerical experiment to compare and analyze the practical crossover points of the protocols. The result of the experiment is presented in Fig. 2. The total number of exponentiations in divide and conquer DHKE protocol is fewer than those of ING, BD, GDH.1 and GDH.2 protocols after $N = 19$ for GDH.1 and GDH.2, and $N = 8$ for BD protocol, as stated in Theorem 2 and Theorem 3.

Theorem 2 $2^{\lceil \log_2 N \rceil}(\lceil \log_2 N \rceil + 1) < N(N+1)$ *for all* $N \geq 8$.

Proof: The proof of Theorem 2 can be obtained using mathematical induction on N. □

Theorem 3 $2^{\lceil \log_2 (N+1) \rceil}(\lceil \log_2 (N+1) \rceil + 1) < \frac{(N+3)N}{2} - 1$ *for all* $N \geq 19$.

Proof: The proof of Theorem 3 can be obtained using mathematical induction on N. □

The characteristics of our protocol are as follows.

1) Total transmission phases: $\lceil \log_2 N \rceil + 1$.
2) Total messages transmitted per P_i: 2 for $P_{N/2-1}$ and P_{N-1}, 1 for others in r-th phase with $1 \leq r \leq \log_2 N - 1$. 1 for all P_i in $(\log_2 N)$-th phase. 0 for all P_i in $(\log_2 N + 1)$-th phase. Recall that in

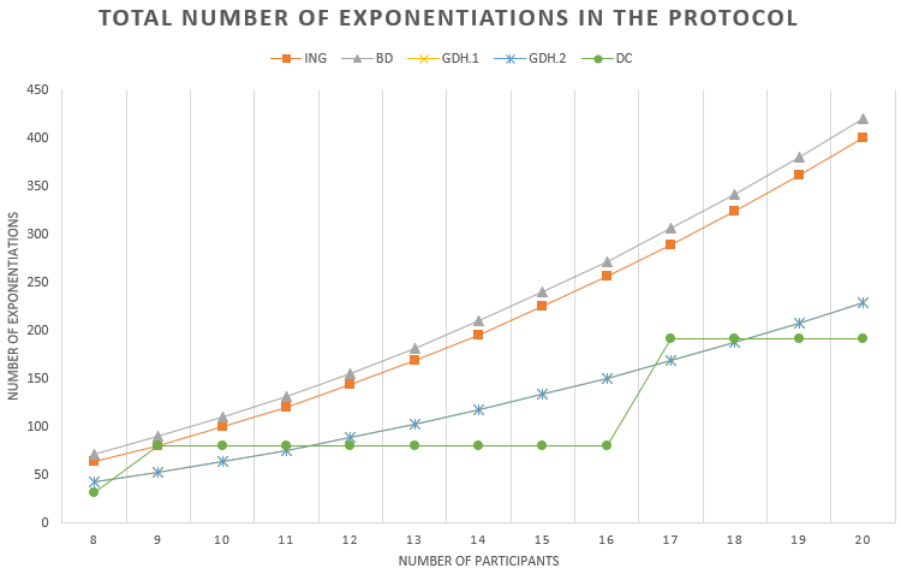

Fig. 2. Total exponentiations in each protocol.

each phase, the number of participants depends on the number of simultaneous processes.

3) Total messages received per P_i:
   2 for $P_{3N/4}$ and $P_{N/4}$, 1 for others in r-th phase with $1 \leq r \leq \log_2 N - 1$. 1 for all P_i in $(\log_2 N)$-th phase. 0 for all P_i in $(\log_2 N + 1)$-th phase. Recall that in each phase, the number of participants depends on the number of simultaneous processes.
4) Total exponentiations per P_i:
   $\lceil \log_2 N \rceil + 1$.
5) Total messages in the protocol:
   $(2^{\lceil \log_2 N \rceil} + \lceil \log_2 N \rceil - 1)\lceil \log_2 N \rceil$.
6) Total exponentiations in the protocol:
   $2^{\lceil \log_2 N \rceil}(\lceil \log_2 N \rceil + 1)$.

According to Corollary 1, our algorithm uses a quasilinear amount of exponentiations with respect to the number of participants. In contrast to GDH.1, GDH.2, and GDH.3, the number of transmission phases in the DC protocol depends on the number of participants. However, the total number of exponentiations in our protocol is fewer than those of GDH.1 and GDH.2. Although the total number of exponentiations is still greater than those in GDH.3 protocol, the DC approach has several advantages in terms of computational burden per participants. According to Lemma 1, each participant in DC protocol performs the same amount of exponentiations. This condition does not happen in GDH.3. By observation, the $(N-1)$-th participant in GDH.3 performs N exponentiations, while the other participants only perform three exponentiations each. Moreover, all members in DC protocol obtain the mutual key simultaneously, while in GDH.3 scheme, the $(N-1)$-th participant retrieves the mutual key before the other participants.

In terms of message transmissions, each participant in DC protocol sends more messages than in GDH.1, GDH.2, and GDH.3 protocols. However, the message size of our protocol is uniform in each transmission. This circumstance does not happen for the protocols proposed in [6], because some participants transmit several values in arrays of different size. The comparison of the computational aspects of these protocol is summarized in Table VI.

D. Security of The Protocol

Security is an essential requirement of a key exchange protocol. The protocol must ensure the secrecy of the mutual key, i.e., no outsider can retrieve the key based on the transmitted messages in the protocol. For N participants, the mutual key of our DC protocol is g^x where $x = \prod_{k=0}^{N-1} \alpha_k$. The security of our protocol is related to the following group Diffie-Hellman problem.

Definition 2 Let $g \in \mathbb{F}_p^*$ be a generator of the finite field $\mathbb{F}_p$ and $S = \{\alpha_i : i \in \{0, 1, \ldots, N-1\}\}$ be a set of secret exponents in group key exchange. The group Diffie-Hellman problem is the problem of computing the value g^x with $x = \prod_{i=0}^{N-1} \alpha_i$ from several values of $g^{y_1}, g^{y_2}, \ldots, g^{y_m}$ for some $m \in \mathbb{N}$ with $g^{y_j} = \prod_{\alpha \in S_j} \alpha$ and S_j is a non-empty proper subset of S.

The group Diffie-Hellman problem is currently intractable provided that the ordinary Diffie-Hellman problem is intractable, i.e., it is difficult to determine the value of g^{xy} given the value of g^x and g^y. Therefore, the group Diffie-Hellman key exchange is as secure as its two-party protocol. Currently, the known way to solve the group Diffie-Hellman problem is by solving several instances of discrete logarithm problem [6, pp. 32]. The reader may refer to [6] for more formal proof regarding this argument.

Our DC protocol is secure due to the the intractability of the group Diffie-Hellman problem. Furthermore, we observe

TABLE V. DIVIDE AND CONQUER APPROACH FOR MULTI-PARTY MEGRELISHVILI PROTOCOL.

Public Parameter Setup
All participants agree on a finite field $\mathbb{F}_p$, an integer n, an $n \times n$ matrix $\mathbf{M}$ over $\mathbb{F}_p$, and a vector $\mathbf{v} \in \mathbb{F}_p^n$.
Private Number Generation
For all $i \in [0, N-1]$, P_i chooses an integer α_i and calculates $\mathbf{P}_i = \mathbf{M}^{\alpha_i}$ as its secret matrix.
Number of Participants Checking
if $N \neq 2^k$ where $k \in \mathbb{Z}$ then Add $2^{\lceil \log_2 N \rceil} - N$ dummies among the existing participants.
Phase 1
Initialization
All participants are divided into two groups, each group consists of $N/2$ participants. P_0 computes $\mathbf{vP}_0$ and sends it to P_1. $P_{N/2}$ also computes $\mathbf{vP}_{N/2}$ and sends it to $P_{(N/2+1) \text{ rem } N}$.
For all $1 \leq i \leq N/2 - 1$ and $N/2 + 1 \leq i \leq N - 1$
P_i right multiplies the value received from P_{i-1} with $\mathbf{P}_i$. If $i = N/2 - 1$ or $i = N - 1$: $P_{N/2-1}$ transmits the result to $P_{N/2}$ and $P_{3N/4}$. P_{N-1} transmits the result to P_0 and $P_{N/4}$. Else: P_i transmits the result to P_{i+1}.
r-th Phase, with $2 \leq r \leq \log_2 N$
Each group from the previous phase performs the same process simultaneously. Thus, in each phase r, there are 2^{r-1} parallel processes. The vectors used is the values received from the $(r-1)$-th phase.
$(\log_2 N + 1)$-th Phase
The number of participants on each process is equal to one. To get the mutual key, each participant P_i right multiplies the value received from the previous phase with $\mathbf{P}_i$.

that the value of the mutual key, i.e., g^x with $x = \prod_{k=0}^{N-1} \alpha_k$ is never transmitted in the protocol. Consequently, our protocol is secure against a passive attack. However, our protocol is not entirely secure against active attack, e.g., man-in-the-middle attack. To overcome this issue, our protocol needs an authentication scheme to prevent such attack.

V. ADAPTATION FOR MEGRELISHVILI KEY EXCHANGE SCHEME

Megrelishvili protocol was first introduced by R. Megrelishvili, M. Chelidze, and K. Chelidze in [9]. This protocol combines matrix exponentiations and vector-matrix multiplications in its implementation. The construction of Megrelishvili protocol for group key distribution have been discussed in [8], [10]. In the previous sections, we have discussed the DC approach for group DHKE protocol. On this chapter, we show that the DC strategy can be adapted for other existing key exchange scheme. As an example, we provide an adaptation of the DC approach for Megrelishvili protocol.

Megrelishvili protocol is a recently developed key exchange scheme that relies on linear algebra operations, i.e., matrix exponentiation and vector-matrix multiplication. As a result, unlike the DHKE, the security of this protocol does not relate to the conventional discrete logarithm problem. This protocol provides an alternative method of key exchange between several participants [8]. We summarize the DC Megrelishvili scheme in Table V.

VI. CONCLUSION AND FUTURE WORKS

In this paper, we have discussed the generalization of the DC scheme for group DHKE protocol. We provide the correctness analysis of the protocol by mathematical induction in Theorem 1. Additionally, Corollary 1 stipulates that the total number of exponentiations in the entire protocol for N participants is $\mathcal{O}(N \log_2 N)$, which means that this protocol is more efficient than ING, BD, GDH.1, and GDH.2 protocols. Our DC protocol requires more exponentiations than GDH.3 protocol presented in [6]. However, our DC protocol has several advantages. Firstly, unlike the participants in GDH.3, each participant in DC protocol performs the same amount of exponentiations. Moreover, all members in DC protocol obtain the mutual key simultaneously. Lastly, the message size of our protocol is uniform for each transmission. In addition to these advantages, we have shown that the DC approach can be adapted in another key exchange protocol, i.e., Megrelishvili protocol. Nevertheless, in this paper, we do not rigorously formalize nor model the message transmissions among participants.

We have discussed the security of Divide and Conquer scheme for DHKE protocol against passive attacks. We argue that the DC DHKE protocol is as secure as its two-party DHKE, provided that the Diffie-Hellman problem is intractable. However, more research regarding the security of the protocol against active attacks still needs to be conducted, especially the one that uses authentication schemes. Moreover, the formalization of the protocol as well as its security analysis using formal framework (e.g.: pi-calculus, Burrows-Abadi-Needham Logic, temporal logic, or Dolev-Yao model) is also needed to ensure the correctness and security of the protocol in a more formal manner.

REFERENCES

[1] Z. Liehuang, L. Lejian, L. Wenzhuo, and Z. Zijian, "An Authenticated Constant Round Group Key Agreement Protocol Based on Elliptic Curve Cryptography," IJCSNS International Journal of Computer Science and Network Security, vol. 6, no. 8 B, pp. 131–134, 2006.

TABLE VI. PROTOCOLS COMPARISON.

Criteria	**ING**	**BD**	**GDH.1**	**GDH.2**	**GDH.3**	**DC**
Total transmission phases	$N-1$	3	2	2	4	$\lceil \log_2 N \rceil + 1$
Messages sent per P_i	$N-1$	2	2 or 1	1	2	1 or 2 per phase
Messages received per P_i	$N-1$	$N-1$	2 or 1	2 or 1	3 or N	1 or 2 per phase
Total messages	$N(N-1)$	$2N-1$	$2(N-1)$	N	$2N-1$	$(N+\log_2 N-1)\log_2 N$
Exponentiations per P_i	N	$N+1$	$i+1$	$i+1$	4 or 2 or $N-1$	$\lceil \log_2 N \rceil + 1$
Total Exponentiations	N^2	$N(N+1)$	$\frac{(N+3)N}{2}-1$	$\frac{(N+3)N}{2}-1$	$5N-6$	$2^{\lceil \log_2 N \rceil}(\lceil \log_2 N \rceil + 1)$
Computational burden per P_i	Homogeneous	Homogeneous	Non-Homogeneous	Non-Homogeneous	Non-Homogeneous	Homogeneous
A *priori* synchronization	Y	Y	N	N	N	Y
Regular DH Key	Y	N	Y	Y	Y	Y
Simultaneous key retrieval per P_i	Y	Y	N	N	N	Y

[2] P. Jaiswal, A. Kumar, and S. Tripathi, "Design of Secure Group Key Agreement Protocol using Elliptic Curve Cryptography," in High Performance Computing and Applications (ICHPCA), 2014 International Conference on. IEEE, 2014, pp. 1–6.

[3] W. Diffie and M. E. Hellman, "New directions in cryptography," Information Theory, IEEE Transactions on, vol. 22, no. 6, pp. 644–654, 1976.

[4] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in Workshop on the Theory and Application of of Cryptographic Techniques. Springer, 1994, pp. 275–286.

[5] I. Ingemarsson, D. T. Tang, and C. K. Wong, "A conference key distribution system," IEEE Transactions on Information theory, vol. 28, no. 5, pp. 714–720, 1982.

[6] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," in Proceedings of the 3rd ACM conference on Computer and communications security. ACM, 1996, pp. 31–37.

[7] S. A. Gaonkar and H. M. Pai, "Extension of Diffie Hellman Algorithm for Multiple Participants," Internationnal Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, pp. 42–47, 2015.

[8] M. Arzaki, "On the Generalizations of Megrelishvili Protocol for Group Key Distribution," Indonesian Journal on Computing (Indo-JC), vol. 2, no. 2, pp. 55–78, 2017.

[9] R. Megrelishvili, M. Chelidze, and K. Chelidze, "On the construction of secret and public-key cryptosystems," Applied Mathematics, Informatics and Mechanics (AMIM), vol. 11, no. 2, pp. 29–36, 2006.

[10] M. Arzaki and B. A. Wahyudi, "Extending Megrelishvili protocol for multi-party key agreement," in 2017 5th International Conference on Information and Communication Technology (ICoICT). IEEE, 2017, pp. 290–297.