Elementary Analysis of the Communication Complexity of Divide-and-Conquer Diffie-Hellman Key Agreement Protocol

Muhammad Arzaki

***Abstract*—We present a rigorous elementary analysis of the communication complexity of the Divide-and-Conquer Diffie-Hellman Key Agreement Protocol (DC-DHKA). The analysis is conducted by first determining the number of transmissions in DC-DHKA and then comparing the resulting communication complexity of this protocol with other variants of Diffie-Hellman key agreement protocols that utilize regular Diffie-Hellman key, namely the ING, GDH.1, GDH.2, and GDH.3 protocols. The mathematical and numerical analyses show that the total number of bits transmitted in the DC-DHKA protocol is always fewer than those of ING, GDH.1, and GDH.2 protocols for a group of $N \geq 19$ participants. In addition, we also prove that the total number of bits required for the entire messages' transmissions in DC-DHKA protocol for N participants that uses the multiplicative group $\mathbb{F}_q^*$ is $\lceil \log_2 q \rceil \cdot \left[2^{\lceil \log_2 N \rceil}(\lceil \log_2 N \rceil + 1) - 2\right]$.**

***Index Terms*—communication complexity, DC-DHKA, Diffie-Hellman, key agreement.**

I. Introduction

Electronic and digital messages are now ubiquitously used as communication mediums, which are often transmitted using the Internet and may contain confidential information. Consequently, messages are usually secured using either a symmetric or an asymmetric cryptosystem. In many cases, symmetric encryption is more preferred because it is generally faster than the asymmetric one [1], [2]. In a symmetric cryptosystem, encryption and decryption are performed using the same key. When the message transmission involves more than two communicating parties, the key management protocol is indispensable.

There are three fundamentally different approaches to group key management protocols [3], [4]. The first one depends on a single entity called a *trusted third party* (TTP) to generate and distribute the key to all group members. This method has a drawback because it requires the key to be sent using a secure channel and the TTP must be constantly available during the entire process of key distribution. The second approach is called the *decentralized group distribution*, which involves a dynamic selection of a subset of group members to generate and distribute keys to another subset of communicating parties. Although this method does not require a single entity like the TTP, it still needs a considerable cost to construct a secure communication channel. The last approach is the contributory group key agreement protocol. Unlike the previous two approaches, this method requires every participant to contribute an equal portion to the mutual group key. Such a key is computed as a function of all members' shares. The term *agreement* is used instead of *distribution* to emphasize the contributory character of the key management. In this method, all participants somewhat agree with the mutual secret key.

Muhammad Arzaki is with the Computing Laboratory, School of Computing, Telkom University, Bandung 40257, Indonesia; email: arzaki@telkomuniversity.ac.id.

Manuscript received June 23, 2022; accepted August 3, 2022.

Since the development of the two-party key agreement protocol by W. Diffie and M. Hellman [5], numerous solutions have been offered to address the multi-party (group) key agreement. Some of these methods are those proposed by Ingemarsson et al. [6], Burmester and Desmedt [7], Steiner et al. [8], Becker and Wille [9], Kim et al. [3], Gaonkar and Pai [10], and Dewoprabowo et al. [11]. The protocols discussed by Ingemarsson (hereinafter referred to as ING protocol), Steiner et al. (hereinafter referred to as GDH protocol), and Dewoparabowo et al. (hereinafter referred to as DC-DHKA) are particularly interesting to investigate because the resulting mutual keys in these protocols are regular Diffie-Hellman key (regular DH key). A regular DH key is a key of the form g^e where g is the generator of a particular finite multiplicative group and e is the product of all secret exponents of all communicating parties.

A comparative analysis of computation and communication costs among ING, BD, GDH.1, GDH.2, GDH.3, and DC-DHKA protocols have been discussed in [11]. From this analysis, the number of modular exponentiations in the DC-DHKA protocol is asymptotically fewer than those of ING, BD, GDH.1, and GDH.2 protocols, but it is asymptotically more than that of GDH.3. The communication complexity analysis in [11] discusses the total number of transmission phases, the total number of messages sent and received per participant, and the types of transmission phases that occurred in the entire protocol. Analysis in [11] is also supplemented with a discussion about the homogeneity of the computational burden for each participant in all relevant protocols. Nevertheless, this analysis does not discuss an elementary aspect regarding the number of bits required to be communicated for constructing the mutual secret key. Although the key agreement protocols that use regular DH keys are not the most efficient ones, theoretical exploration related to the elementary analysis of communication complexity for these protocols is important and interesting to investigate.

Our objective is to investigate the elementary communication complexity of the DC-DHKA protocol. Here, the term elementary refers to the investigation regarding the number of bits required to be transmitted for constructing the mutual secret key. To do so, we discuss the total number of transmissions in the protocol as well as the size of each message transmitted. We perform both mathematical analysis and numerical experiments to investigate the elementary communication complexity of DC-DHKA protocols. For comparative purposes, we also present similar investigations on ING, GDH.1, GDH.2, and GDH.3 protocols to derive a conclusion regarding the elementary communication complexities of pertinent key agreement protocols that use the regular DH key. Thus, our analysis complements the previous evaluation of the DC-DHKA protocol regarding its complexity and security in [11], [12].

II. Communication Complexity of Key Agreement Protocols

Communication complexity is a mathematical study that addresses the amount of communication required by the communicating parties to achieve a common goal [13]. It was first introduced by Yao [14] and it is also defined as the minimum number of bits that the communicating parties need to exchange (or transmit) for achieving a particular objective [15]. The notion of communication complexity has been used to determine the efficiency of communication in key agreement protocols as discussed in [9]. The discussion regarding the communication complexity of key agreement protocols is also recently addressed by Haitner et al. [16].

Analyses regarding the communication complexity of Diffie-Hellman-based group key agreement protocols are discussed by Steiner et al. [8], Becker and Wille [9], and Dewoprabowo et al. [11]. In [8], [9], [11], the measurement of the communication complexity is performed by considering the number of messages transmitted in the entire protocol, the number of exchanges or transmissions, and the type of phases as well as the number of phases required in the transmission.

Assuming that a broadcast transmission (a transmission of a message from one participant to all other participants) is not allowed, it is proven that for any key agreement protocol involving $N \geq 4$ participants, the total number of messages is bounded below by $2(N-2)$, the total number of exchanges is bounded below by $2(N-4)$, and the total number of simple rounds (a round in which every party sends and receives at most one message) is bounded below by $\lceil \log_2 N \rceil$ [9]. In contrast, if the broadcast transmission is allowed, then the total number of messages, as well as the number of exchanges in the protocol, is bounded below by N. Some examples of key agreement protocols that achieve the aforementioned efficiency are the *octopus* and *hypercube* protocols described in [9] as well as the tree-based group Diffie-Hellman protocol (TGDH) described in [3]. However, it should be noted that all of these protocols do not use the regular DH key as their mutual secret key.

Although the communication complexity of group key agreements that employ the regular DH key has been discussed in [8], [9], [11], all of its investigations do not address the number of bits required in the transmission process to obtain the mutual secret key. In this paper, we conduct elementary analyses of the communication complexity of group key agreements by considering the number of bits used in the entire protocol. Hence, even though two protocols might have different quantity in terms of the number of transmissions, they can have an identical number of bits that needs to be transmitted to agree on a mutual secret key.

III. Related Diffie-Hellman Key Agreement Protocols and Their Communication Complexities

In this section, we perform elementary analyses of the communication complexities of several Diffie-Hellman key agreement protocols that utilize the regular Diffie-Hellman key (DH key), namely the ING, GDH.1, GDH.2, and GDH.3 protocols, for comparative purposes. We begin our analyses by introducing some mathematical notations and definitions we use throughout this paper which are adapted from previous works such as [8], [11], [12]. These theoretical assumptions are used to analyze the related Diffie-Hellman key agreement protocols as well as the DC-DHKA.

A. Mathematical Preliminaries and Assumptions

Throughout this paper—unless it is specified otherwise—a key agreement protocol is always observed in a collection of N participants, denoted by $M_0, M_1, \ldots, M_{N-1}$, where $N > 1$ is an integer. In practice, these participants can be any communicating parties. All computations in a protocol are observed in the multiplicative group of a finite field $\mathbb{F}_q$ which is denoted by $\mathbb{F}_q^*$. Here, we have $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$. We denote a generator of $\mathbb{F}_q^*$ by g, i.e., every $a \in \mathbb{F}_q^*$ can be expressed as $a = g^y$ for some integer $0 \leq y \leq q-1$.

Although in algebra the finite field $\mathbb{F}_q$ is well-defined as long as q is a prime power, our investigation generally assumes that q is a prime number. This assumption is used in conformity with the underlying theories in [8], [11], [12]. For any two elements $a, b \in \mathbb{F}_q$, we write $c = a \bmod b$ if $a = \alpha \cdot b + c$ for some $\alpha \in \mathbb{F}_q$ and $0 \leq c \leq b-1$. In this paper, we use the mod operator in $\mathbb{F}_q$ interchangeably with the usual mod operator in the set of integers. The semantics of this operator is referred from the context of a mathematical expression. Furthermore, since $\mathbb{F}_q$ is a field, the value $a^{-1} \bmod q \in \mathbb{F}_q$ for any non-zero a refers to the multiplicative inverse of a, that is $a \cdot (a^{-1} \bmod q) = 1$ in $\mathbb{F}_q$. When q is a prime number, $a^{-1} \bmod q$ can be easily determined using the extended Euclid's algorithm (see, e.g., [17, Theorem 1.11] and [18, p. 286]).

Message transmissions in ING, GDH.1, GDH.2, GDH.3, and DC-DHKA protocols involve sending at least one value of an expression g^x where x is a positive integer and g is a fixed and publicly known generator of $\mathbb{F}_q^*$. Since $g^x \in \mathbb{F}_q^*$ and every element of $\mathbb{F}_q$ can be represented using $\lceil \log_2 q \rceil$ bits, then we assume that any message of the form g^x requires a data representation of $\lceil \log_2 q \rceil$ bits.

Every participant M_i for $0 \leq i \leq N-1$ possesses a secret exponent s_i. Theoretically, the value of s_i is unknown to anyone except M_i. We generally assume that $1 \leq s_i \leq q-1$. Although in practice the value of s_i can be any positive integer provided that $s_i \not\equiv 0 \pmod{q}$, the value of g^{s_i} is always restricted to the set $\mathbb{F}_q^*$. Moreover, notice that the condition $s_i \not\equiv 0 (\bmod q)$ ensures that s_i is invertible in $\mathbb{F}_q^*$ and $g^{s_i} \in \mathbb{F}_q^*$.

Suppose we consider a sequence of positive integers $\alpha_i, \alpha_{i+1}, \ldots, \alpha_{j-1}, \alpha_j$. For two integers i and j such that $i \leq j$, we define $\prod_{k=i}^{j} \alpha_k = \alpha_i \alpha_{i+1} \cdots \alpha_{j-1} \alpha_j$. We also define $\prod_{k=i}^{i} \alpha_k = \alpha_i$ and if $i > j$ we define the empty product $\prod_{k=i}^{j} \alpha_k = 1$.

In a Diffie-Hellman key agreement with N participants $M_0, M_1, \ldots, M_{N-1}$ whose secret exponents are respectively $s_0, s_1, \ldots, s_{N-1}$, we define the regular Diffie-Hellman key (regular DH key) as the value g^e where $e = \prod_{i=0}^{N-1} s_i$. The security of a DHKA protocol is related to the group Diffie-Hellman problem, that is, we assume that it is currently computationally intractable to find the value of g^e using one or more values of the form g^y where $y = \prod_{s \in S'} s$ where $S' \subset S$ and $S = \{s_0, s_1, \ldots, s_{N-1}\}$ [11, Definition 2]. This mathematical problem is generalized from the conventional two-party Diffie-Hellman problem in [17, p. 69].

Some key agreement protocols use list data structure to store one or more values over $\mathbb{F}_q^*$. Throughout this paper, we consider the zero-based index convention for list. A list L of length n over $\mathbb{F}_q^*$ is denoted as $L = \big[L[0], L[1], \ldots, L[n-1]\big]$ where $L[i] \in \mathbb{F}_q^*$ for each i such that $0 \leq i \leq n-1$.

B. The ING Protocol and Its Analysis

The ING[1] protocol was first formalized by Ingemarsson et al. in [6]. Suppose there are N participants, labeled as $M_0, M_1, \ldots, M_{N-1}$, and each participant M_i has a secret exponent s_i. The common secret key of this protocol is g^e where $e = \prod_{i=0}^{N-1} s_i$. To obtain this key, the protocol requires $N-1$ phases of message transmissions.

The ING protocol works in a straightforward way. To begin with, the participants are arranged in a somewhat circular configuration. For simplicity, suppose the message sent by M_i in the phase j where $0 \leq i \leq N-1$ and $0 \leq j \leq N-2$ is denoted by $\mathsf{Message}(i, j)$. Before any transmission takes place, every participant M_i for $0 \leq i \leq N-1$ computes $\mathsf{Message}(i, 0) = g^{s_i}$. In the initial phase of transmission, M_i sends $\mathsf{Message}(i, 0)$ to $M_{(i+1) \bmod N}$. Once receiving this message, $M_{(i+1) \bmod N}$ computes $\mathsf{Message}((i+1) \bmod N, 1) = \mathsf{Message}(i \bmod N, 0)^{s_{i+1}}$ and then sends this value to $M_{(i+2) \bmod N}$. In general, in phase j where $1 \leq j \leq N-2$, every participant M_i for $0 \leq i \leq N-1$ computes $\mathsf{Message}((i-1) \bmod N, j-1)^{s_i}$ and subsequently sends the result to $M_{(i+1) \bmod N}$. In the last phase (phase $N-1$), every participant M_i calculates $\mathsf{Message}((i-1) \bmod N, N-2)^{s_i}$. This value is not sent anywhere but it is used as the common secret key. Using mathematical induction, we can prove that at the end of phase $N-1$ every participant M_i $(0 \leq i \leq N-1)$ obtains $\mathsf{Message}((i-1) \bmod N, N-2)^{s_i} = g^e$, where $e = \prod_{i=0}^{N-1} s_i$. In addition, we observe that every participant in this protocol eventually obtains the common secret key at the same time.

Notice that ING protocol involves $N-1$ phases of transmissions (namely phase 0 to $N-2$) and a message sent from M_i to $M_{(i+1) \bmod N}$ is of the form g^x where $x = \prod_{s \in S'} s$ and S' is a proper subset of the set of secret exponents $\{s_0, s_1, \ldots, s_{N-1}\}$. Using the previous assumption that any value of the form g^x requires $\lceil \log_2 q \rceil$ bits of data representation, then the total number of bits involved in the entire transmissions in ING protocol with N participants is $\lceil \log_2 q \rceil \cdot N \cdot (N-1)$.

[1]The original name of this protocol is CDKS (Conference Key Distribution System). The term ING was popularized in [8].

C. The GDH.1 Protocol and Its Analysis

The GDH.1 protocol was one of the first non-trivial Diffie-Hellman key agreement protocols that use a regular DH key.[2] This protocol was first introduced by Steiner et al. in [8]. It uses two main phases, namely the *upflow* and *downflow* phases. Unlike the ING protocol in which the participants are arranged in a somewhat circular configuration, in GDH.1 protocol the communicating parties are set up in a linear configuration. In both upflow and downflow phases, a message is a list whose entries are elements of $\mathbb{F}_q^*$.

At the beginning of the upflow phase, M_0 defines a list $\mathsf{U}_0 = [g^{s_0}]$ and sends this list to M_1. For each i such that $1 \leq i \leq N-2$, the participant M_i constructs a list U_i using the following procedure:

1) M_i takes the last entry of the list U_{i-1} and raises it to the power of s_i, suppose we denote this operation by $(\mathsf{U}_{i-1}[last])^{s_i}$;
2) M_i defines the list $\mathsf{U}_i = \mathsf{U}_{i-1} \parallel [(\mathsf{U}_{i-1}[last]^{s_i})]$, where $\parallel$ represents the concatenation operator for two lists.

At the end of the upflow phase, M_{N-1} takes the last entry of U_{N-2} and raises it to the power of s_{N-1}. We can easily prove that the result of this calculation, namely $(\mathsf{U}_{N-2}[last])^{s_{N-1}}$, is the common secret key (the regular DH key). Here, M_{N-1} obtains the common secret key before any other participant.

To distribute the key to other participants, M_{N-1} initializes a downflow list D_{N-1} of length $N-1$ as

$$\begin{aligned} \mathsf{D}_{N-1} &= \left[g^{\left(\prod_{k=N-1}^{N-1} s_k\right)\left(\prod_{k=0}^{j-1} s_k\right)} : 0 \leq j \leq N-2 \right] \\ &= \left[g^{s_{N-1}}, g^{s_{N-1}s_0}, \ldots, g^{s_{N-1}s_0 s_1 \cdots s_{N-4} s_{N-3}} \right]. \end{aligned} \tag{1}$$

Each participant M_i for $0 \leq i \leq N-2$ in descending order (starting from $N-2$) performs the following steps:

1) M_i takes the last entry of D_{i+1} and raises it to the power of s_i to obtain the common secret key;
2) if $i \neq 0$, M_i defines D_i by removing the last entry of D_{i+1} and then raising all remaining entries to the power of s_i, this list is then sent to M_{i-1}.

Observe that every participant M_i for $1 \leq i \leq N-2$ obtains the list D_{i+1} from M_{i+1} of the form

$$\mathsf{D}_{i+1} = \left[g^{\left(\prod_{k=i+1}^{N-1} s_k\right)\left(\prod_{k=0}^{j-1} s_k\right)} : 0 \leq j \leq i \right]. \tag{2}$$

To acquire the common secret key M_i needs to take the last entry of D_{i+1}, namely $g^{\left(\prod_{k=i+1}^{N-1} s_k\right)\left(\prod_{k=0}^{i-1} s_k\right)}$, and raises it to the power of s_i. Notice that for $i = 0$, M_0 obtains $\mathsf{D}_1 = \left[g^{\left(\prod_{k=1}^{N-1} s_k\right)\left(\prod_{k=0}^{j-1} s_k\right)} : 0 \leq j \leq 0 \right] = \left[g^{\prod_{k=1}^{N-1} s_k} \right]$.

[2]The term GDH is an abbreviation of *Group Diffie-Hellman*.

Thus, to obtain the common secret key M_0 needs to raise the only element in D_1 to the power of s_0. The correctness of this protocol relies on the fact that the exponent product, i.e., the value $\prod_{k\in[0,N-1]} s_k$, is always identical regardless of the order of the exponents. This happens due to the commutativity of integer multiplication.

From the description of this protocol, we observe that the participant M_i is the $(N-i)$-th participant to retrieve the mutual secret key. That is, the first group member to obtain the key is M_{N-1}, followed by M_{N-2}, and so on until M_0. Hence, unlike the ING protocol, every participant in GDH.1 obtains the common secret key at different times.

To analyze the number of bits required in the entire transmission of GDH.1 we assume that every value of the form g^x where x is a positive integer requires a data representation of $\lceil \log_2 q\rceil$ bits. Notice that during the upflow phase every participant M_i for $0 \leq i \leq N-2$ sends a list U_i of length $i+1$ whose entries are of the form g^x for some positive integer x. As a result, the total number of bits required to send all messages during the upflow phase is given by

$$\begin{aligned}\sum_{i=0}^{N-2}(i+1)\lceil \log_2 q\rceil &= \lceil \log_2 q\rceil \cdot \sum_{i=1}^{N-1} i \\ &= \lceil \log_2 q\rceil \cdot \frac{N\cdot(N-1)}{2}. \qquad (3)\end{aligned}$$

For the downflow phase, each participant M_i for $1 \leq i \leq N-1$ in descending order sends a list D_i of length i to M_{i-1}. Every entry of D_i is of the form g^x for some positive integer x and requires $\lceil \log_2 q\rceil$ bits of data representation. As a result, the total number of bits required to send all messages during the downflow phase is given by

$$\sum_{i=1}^{N-1} i\cdot\lceil \log_2 q\rceil = \lceil \log_2 q\rceil \cdot \frac{N\cdot(N-1)}{2}. \qquad (4)$$

Finally, by combining (3) and (4) we infer that the total number of bits required to represent the transmissions in the GDH.1 protocol is $\lceil \log_2 q\rceil \cdot N\cdot(N-1)$, which is identical to the number of bits required in the entire transmissions of ING protocol described in Section III-B.

D. The GDH.2 Protocol and Its Analysis

The GDH.2 protocol was first formalized by Steiner et al. and it uses two phases, namely the upflow and broadcast phases [8]. As in GDH.1 protocol, the objective of the upflow phase is to gather the contribution of each participant except M_{N-1}. However, unlike GDH.1 protocol in which the participants acquire the common secret key at different times, in GDH.2 every participant apart from M_{N-1} gets the common secret key simultaneously.

During the upflow phase, all participants are set up in a linear configuration as in the upflow phase in GDH.1. Nevertheless, the computations performed in the upflow phase in GDH.2 are different from those calculations carried out in the upflow phase in GDH.1. Initially participant M_0 defines a list $\mathsf{U}_0 = \left[g^{s_0}\right]$ and sends this list to M_1. Once receiving U_0 from M_0, M_1 defines a list $\mathsf{U}_1 = \left[g^{s_0 s_1}, g^{s_0}, g^{s_1}\right]$ and sends this list to M_2. Afterward, for each i such that $2 \leq i \leq N-2$, participant M_i receives U_{i-1} of length $i+1$ from M_{i-1} and construct a list U_i of $i+2$ entries defined as follows:

1) $\mathsf{U}_i[0] = (\mathsf{U}_{i-1}[0])^{s_i}$,
2) $\mathsf{U}_i[1] = \mathsf{U}_{i-1}[0]$, and
3) $\mathsf{U}_i[j] = \mathsf{U}_{i-1}[j-1]$ for all j satisfying $2 \leq j \leq i+1$.

Notice that for all i such that $2 \leq i \leq N-1$ we have $\mathsf{U}_{i-1} = \left[g^{\prod_{k=0}^{i-1} s_k}, g^{\prod_{k=0,k\neq i-1}^{i-1} s_k}, g^{\prod_{k=0,k\neq i-2}^{i-1} s_k}, \ldots, g^{\prod_{k=0,k\neq 0}^{i-1} s_k}\right]$. At the end of the upflow phase M_{N-1} obtains the common secret key by taking the first element of U_{N-2} and raising it to the power of s_{N-1}, i.e., calculating $(\mathsf{U}_{N-2}[0])^{s_{N-1}}$.

Unlike the GDH.1 protocol, in GDH.2 all participants apart from M_{N-1} can obtain the mutual secret key at the same time. This is possible using the broadcast phase performed by M_{N-1}. Initially, M_{N-1} defines a broadcast list B of length $N-1$ using the upflow list U_{N-2} as follows:

$$\begin{aligned}\mathsf{B}[j] &= (\mathsf{U}_{N-2}[j+1])^{s_{N-1}} \text{ for all } 0 \leq j \leq N-2 \\ \mathsf{B} &= \left[g^{\prod_{k=0,k\neq N-2}^{N-1} s_k}, \ldots, g^{\prod_{k=0,k\neq 0}^{N-1} s_k}\right].\end{aligned} \qquad (5)$$

List B is then broadcasted by M_{N-1} to all other participants. To obtain the common secret key, the participant M_i needs to read $\mathsf{B}[N-2-i]$ and then raises it to the power of s_i. Notice that instead of broadcasting the whole list B, M_{N-1} can also directly send a value $\mathsf{B}[N-2-i]$ to M_i. In either way, M_{N-1} needs to send $N-1$ values of the form g^x where x is a positive integer. The correctness of this protocol relies on the commutativity of the exponent product as explained in the correctness of GDH.1 protocol in Section III-C.

To analyze the number of bits required in the communication of GDH.2, we split our analysis for the upflow and broadcast phases. For the upflow phase, initially M_0 sends a list of an element to M_1, namely $\left[g^{s_0}\right]$. Subsequently, M_1 sends a list of three entries to M_2, namely $\left[g^{s_0 s_1}, g^{s_0}, g^{s_1}\right]$. Furthermore, for any i such that $2 \leq i \leq N-2$, M_i sends a list U_i of length $i+2$ as described in the definition of U_i. Hence, the total number of bits required in the entire transmission for the upflow phase is

$$\begin{aligned}&\lceil \log_2 q\rceil + 3\lceil \log_2 q\rceil + \sum_{i=2}^{N-2}\lceil \log_2 q\rceil\cdot(i+2) \\ &= 4\lceil \log_2 q\rceil + \lceil \log_2 q\rceil \sum_{i=4}^{N} i \\ &= 4\lceil \log_2 q\rceil + \lceil \log_2 q\rceil \frac{(N+4)(N-3)}{2} \\ &= \frac{1}{2}\cdot\lceil \log_2 q\rceil\cdot(N^2+N-4) \qquad (6)\end{aligned}$$

For the broadcast phase, observe that M_{N-1} sends a list B of length $N-1$ whose entries are the elements of $\mathbb{F}_q^*$. Consequently, this phase requires $\lceil \log_2 q\rceil \cdot (N-1)$ bits of data representation. By combining this result with (6) the total bits required in the entire transmission of GDH.2 protocol is $\frac{1}{2}\cdot\lceil \log_2 q\rceil\cdot(N^2+3N-6)$.

E. The GDH.3 Protocol and Its Analysis

The GDH.3 protocol uses four phases for constructing the common secret key, namely upflow, response, and two broadcast phases. This protocol is more efficient than GDH.1, GDH.2, and DC-DHKA protocols in terms of exponentiation operations [11], [12]. At the beginning of the upflow phase, M_0 computes g^{s_0} and sends it to M_1. Afterward, every participant M_i for $1 \leq i \leq N-2$ receives $g^{\prod_{k=0}^{i-1} s_k}$ from M_{i-1}, and subsequently raises this value to the power of s_i to produce $g^{\prod_{k=0}^{i} s_k}$, which is then sent to M_{i+1}. At the end of the upflow phase, M_{N-1} obtains $g^{\prod_{k=0}^{N-2} s_k}$ and raises this value to the power of s_{N-1} to get the mutual secret key. Hence, as in the GDH.1 and GDH.2 protocols, participant M_{N-1} in GDH.3 obtains the common secret key before any other participant.

The first broadcast phase is performed by M_{N-2} by sending the value $g^{\prod_{k=0}^{N-2}}$ to every participant except M_{N-1}. Here M_{N-2} sends this value to $N-2$ participants $M_0, M_1, \ldots, M_{N-3}$.

The response phase is carried out by every participant M_i where $0 \leq i \leq N-2$ and its objective is to remove the secret exponent s_i from the value $g^{\prod_{k=0}^{N-2} s_k}$. Here, initially, each participant computes the multiplicative inverse of their secret exponent in modulo q, i.e., s_i^{-1}. Subsequently, each participant uses the value obtained from the first broadcast phase and computes $\mathsf{R}[i] = \left(g^{\prod_{k=0}^{N-2} s_k}\right)^{s_i^{-1}} = g^{\prod_{k=0,k\neq i}^{N-2} s_k}$. The value of $\mathsf{R}[i]$ is then sent to M_{N-1}.

Upon receiving $\mathsf{R}[i]$ from all other participants, M_{N-1} defines the broadcast list B of length $N-1$ such that $\mathsf{B}[j] = (\mathsf{R}[j])^{s_{N-1}} = g^{\prod_{k=0,k\neq j}^{N-1}}$ for all $0 \leq j \leq N-2$. The list B is then broadcasted to all other participants. However, as in the broadcast phase of GDH.2, M_{N-1} can send an individual value of $\mathsf{B}[i]$ instead of the entire list B to M_i for $0 \leq i \leq N-2$. Nevertheless, in either approach, M_{N-1} needs to send $N-1$ values of the form g^x for some positive integer x. To obtain the mutual secret key, every other participant M_i for $0 \leq i \leq N-2$ reads $\mathsf{B}[i] = g^{\prod_{k=0,k\neq i}^{N-1}}$ and then raises this value to the power of s_i to obtain $g^{\prod_{k=0}^{N-1} s_k}$. Notice that as in the GDH.1 and GDH.2 protocols, the correctness of GDH.3 protocol comes from the commutativity of the exponent product.

To determine the number of bits transmitted in the entire communication of GDH.3, we divide our analysis for each of the phases. For the upflow phase, every participant M_i for $0 \leq i \leq N-2$ sends $g^{\prod_{k=0}^{i} s_k}$ to M_{i+1}. Using the assumption in Section III-A, each transmission from M_i to M_{i+1} for $0 \leq i \leq N-2$ requires $\lceil \log_2 q \rceil$ bits of data representation. As a consequence, the entire transmissions in the upflow phase requires $\sum_{i=0}^{N-2} \lceil \log_2 q \rceil = \lceil \log_2 q \rceil \cdot (N-1)$ bits of data representation.

Next, in the first broadcast phase, M_{N-2} transmits $g^{\prod_{k=0}^{N-2} s_k}$ to all other participants except M_{N-1}. Since only one value of the form g^x is transmitted, then this phase requires $\lceil \log_2 q \rceil$ bits of data representation. However, if instead M_{N-2} sends this value individually to every M_i except M_{N-1}, then the total number of bits needed for the transmissions in this phase is $\lceil \log_2 q \rceil \cdot (N-2)$.

The response phase involves $N-1$ independent transmissions of the value $\mathsf{R}[i] = \left(g^{\prod_{k=0}^{N-2} s_k}\right)^{s_i^{-1}} = g^{\prod_{k=0,k\neq i}^{N-2} s_k}$, each value $\mathsf{R}[i]$ is sent by M_i where $0 \leq i \leq N-2$ to M_{N-1}. Thus, the total number of bits transmitted in this phase is $\lceil \log_2 q \rceil \cdot (N-1)$.

The second broadcast phase is performed by M_{N-1} by transmitting a list B of length $N-1$ to every participant M_i where $0 \leq i \leq N-2$. The total number of bits required in this phase is $\lceil \log_2 q \rceil \cdot (N-1)$.

If we assume that the first broadcast phase requires a single transmission of the value $g^{\prod_{k=0}^{N-2} s_k}$, then the total number of bits required in the entire transmissions of GDH.3 protocol is

$$\begin{aligned} &\lceil \log_2 q \rceil \cdot (N-1) + \lceil \log_2 q \rceil + \lceil \log_2 q \rceil \cdot (N-1) \\ &\quad + \lceil \log_2 q \rceil \cdot (N-1) = \lceil \log_2 q \rceil \cdot (3N-2). \end{aligned} \tag{7}$$

However, if we consider that M_{N-2} sends the value $g^{\prod_{k=0}^{N-2} s_k}$ to each M_i for $0 \leq i \leq N-3$ individually, then the total number of bits needed in the entire transmissions of GDH.3 protocol is

$$\begin{aligned} &\lceil \log_2 q \rceil \cdot (N-2) + \lceil \log_2 q \rceil \cdot (N-1) + \lceil \log_2 q \rceil \cdot (N-1) \\ &\quad + \lceil \log_2 q \rceil \cdot (N-1) = \lceil \log_2 q \rceil \cdot (4N-5). \end{aligned} \tag{8}$$

Notice that both (7) and (8) are linear polynomials in N, that is, the total number of bits communicated in GDH.3 is bounded by a constant factor of the number of participants.

IV. DC-DHKA Protocol and Its Elementary Communication Complexity Analysis

The divide-and-conquer approach for the Diffie-Hellman key agreement was first proposed by Gaonkar and Pai [10]. The objective of this approach is to reduce the number of exponentiations performed during the common secret key construction. This protocol is further formalized and generalized by Dewoprabowo et al. [11]. It is theoretically proven that the number of exponentiations in DC-DHKA[3] is asymptotically fewer than those in ING, GDH.1, and GDH.2 protocols. Although the overall number of exponentiations in DC-DHKA is asymptotically greater than those in GDH.3, DC-DHKA outperforms GDH.3 in several aspects. First, every participant in DC-DHKA performs homogeneous mathematical operations and thus the number of computations for each participant are identical [11]. This does not happen in GDH.3 since M_{N-2} and M_{N-1} perform more computations than other participants. Second, in DC-DHKA every communicating party obtains the mutual secret key at the same time [12]. Such a condition does not happen in GDH.1, GDH.2, and GDH.3 protocols. We refer the reader to [11], [12] for more comprehensive references regarding the complexity analysis (in terms of exponentiation operations) and security analysis of the protocol. Nevertheless, to our knowledge, an investigation into the communication complexity of DC-DHKA has not been performed rigorously. Before we discuss the communication complexity of DC-DHKA, we first recall the workflow of the protocol.

[3]DC-DHKA stands for *Divide-and-Conquer-based Diffie-Hellman Key Agreement*.

A. The Workflow of DC-DHKA Protocol

The DC-DHKA protocol works recursively using the divide-and-conquer approach and assumes that the number of communicating parties is not fewer than four. Before any transmission takes place, the communicating parties are arranged in sequential order. Suppose we consider a group of $N \geq 4$ participants, $M_0, M_1, \ldots, M_N$, then the position of each M_i must be fixed before any message is sent.

The original description of DC-DHKA only works if the number of communicating parties is an exact power of two. This is because the number of participants must be split equally in every transition between two consecutive phases. Nevertheless, DC-DHKA can still be implemented if the number of participants is not an exact power of two by inserting some *dummy participants*. This process is called *padding* and it is explained in [11], [12]. The purpose of this padding is to make the number of participants an exact power of two. In general, every dummy participant is located between two non-dummy participants whenever such a condition is possible. Moreover, the secret exponent of every dummy participant is set to 1 to prevent additional unnecessary exponentiations. For example, if we consider a group of five participants $M_0, M_1, \ldots, M_4$, then to perform DC-DHKA we need to add three dummies M_5, M_6, M_7 and the configuration of the group is $M_0, M_5, M_1, M_6, M_2, M_7, M_3, M_4$. Here, the secret exponents of M_5, M_6, and M_7 are set to 1.

Analysis and numerical experiments show that the number of exponentiations in DC-DHKA is fewer than those of ING, GDH.1, and GDH.2 protocols if the number of participants is at least 19 [11]. Henceforth, we typically assume that the number of participants in DC-DHKA is N where N is an exact power of two and $N \geq 19$ except it is specified otherwise.

We first explain the base case of DC-DHKA protocol involving four participants M_0, M_1, M_2, M_3 whose secret exponents are respectively a, b, c, d. The common secret key construction is performed in three phases as follows:

1) In the first phase M_0 computes g^a and sends this value to M_1. At the same time M_2 computes g^c and sends this value M_3. Subsequently, M_1 calculates $(g^a)^b$ and transmits it to M_2 and M_3, while M_3 calculates $(g^c)^d$ and transmits it to M_0 and M_1.
2) In the second phase, M_0 and M_1 that receive g^{cd} from M_3 respectively compute $(g^{cd})^a$ and $(g^{cd})^b$. At the same time, M_2 and M_3 that receive g^{ab} from M_0 respectively compute $(g^{ab})^c$ and $(g^{ab})^d$.
3) In the third phase, M_0 and M_1 exchange their messages, hence M_0 gets g^{cdb} and M_1 gets g^{cda}. In parallel, M_2 and M_3 also exchange their messages, hence M_2 gets g^{abd} and M_3 gets g^{abc}. Observe that to obtain the mutual secret key, all participants raise their values with their own secret exponent, here M_0 computes $(g^{cdb})^a$, M_1 computes $(g^{cda})^b$, M_2 computes $(g^{abd})^c$, and M_3 computes $(g^{abc})^d$.

As in GDH.1, GDH.2, and GDH.3 protocols, the correctness of the DC-DHKA protocol comes from the commutativity of the exponent product. The message sequence chart for the base case of the DC-DHKA protocol is illustrated in Fig. 1.

We can generalize the aforementioned base case of the DC-DHKA protocol for N participants where $N = 2^K$ for any integer $K \geq 3$. Before we discuss the generalization of DC-DHKA for N participants, we first explain the DC-DHKA protocol for eight participants labeled as $M_0, M_1, \ldots, M_7$ whose secret exponents are respectively $a, b, c, d, k, \ell, m, n$. The construction of the mutual secret key is achieved in four phases as follows:

1) The first phase commences when M_0 computes g^a and sends it to M_1, while at the same time M_4 computes g^k and sends it to M_5. After receiving g^a, M_1 computes $(g^a)^b$ and sends it to M_2, while simultaneously M_5 computes $(g^k)^\ell$ and sends it to M_6. Afterward M_2 computes $(g^{ab})^c$ and sends this value to M_3, while concurrently M_6 computes $(g^{k\ell})^m$ and sends it to M_7. At the end of the first phase, M_3 computes $(g^{abc})^d$ and transmits this value to M_4 and M_6, while in parallel M_7 computes $(g^{k\ell m})^n$ and transmits this value to M_0 and M_2. When the first phase is concluded, two new subgroups is formed, namely:
   a) the first subgroup of M_0, M_1, M_2, M_3, where M_0 and M_2 possess the value $g^{k\ell mn}$,
   b) the second subgroup of M_4, M_5, M_6, M_7, where M_4 and M_6 possess the value g^{abcd}.

   In the subsequent phases, each participant in each subgroup does not communicate (send or receive messages) with other participants in another subgroup.
2) In the second phase, there are two parallel processes of computations and transmissions in each subgroup as explained at the end of the first phase. The overall workflow occurring in each subgroup involves four participants and it is analogous to the first phase of the base case for DC-DHKA, namely:
   a) In the first subgroup, participant M_0 computes $(g^{k\ell mn})^a$ and sends it to M_1, while simultaneously M_2 computes $(g^{k\ell mn})^c$ and sends it to M_3. Afterward, M_1 computes $(g^{k\ell mna})^b$ and transmits it to M_2 and M_3, while concurrently M_3 computes $(g^{k\ell mnc})^d$ and transmits it to M_0 and M_1. At the end of this phase M_0 and M_1 have $g^{k\ell mncd}$, whereas M_2 and M_3 have $g^{k\ell mnab}$.
   b) In the second subgroup, participant M_4 computes $(g^{abcd})^k$ and sends it to M_5, while at the same time M_6 computes $(g^{abcd})^m$ and sends it to M_7. After that, M_5 computes $(g^{abcdk})^\ell$ and transmits it to M_6 and M_7, while simultaneously M_7 computes $(g^{abcdm})^n$ and transmits it to M_4 and M_5. At the end of this phase M_4 and M_5 own g^{abcdmn}, whereas M_6 and M_7 own $g^{abcdk\ell}$.
3) In the third phase, each participant raises its own value with its secret exponent and perform a message exchange between two participants, namely: M_0 and M_1, M_2 and M_3, M_4 and M_5, and M_6 and M_7.
4) In the fourth phase, each participant raises the value obtained from the third phase with its secret exponent to obtain the mutual secret key.

The message sequence chart for DC-DHKA of eight partic-

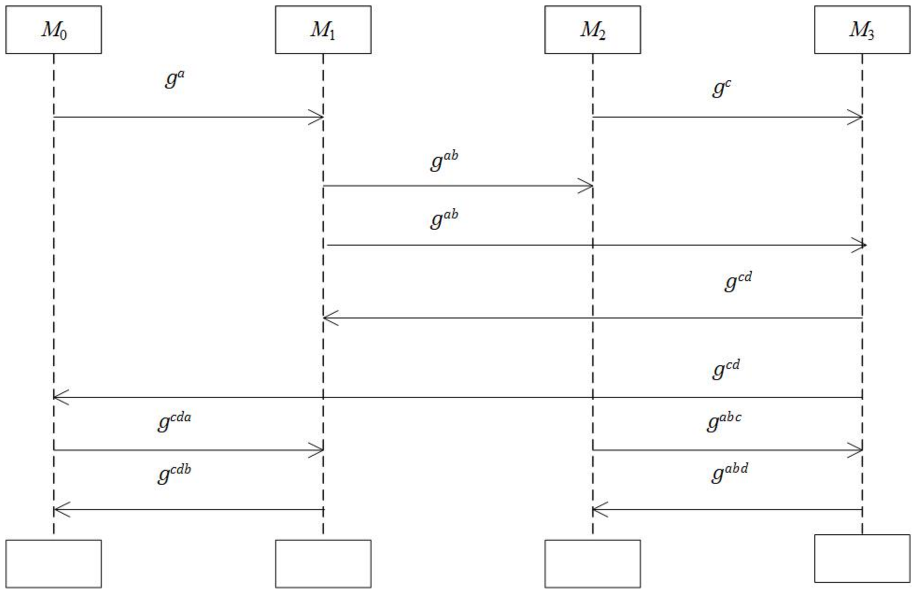

Fig. 1: The message sequence chart for the base case of DC-DHKA protocol as explained in [11], [12]. The participants are M_0, M_1, M_2, M_3 and their corresponding secret exponents are a, b, c, d.

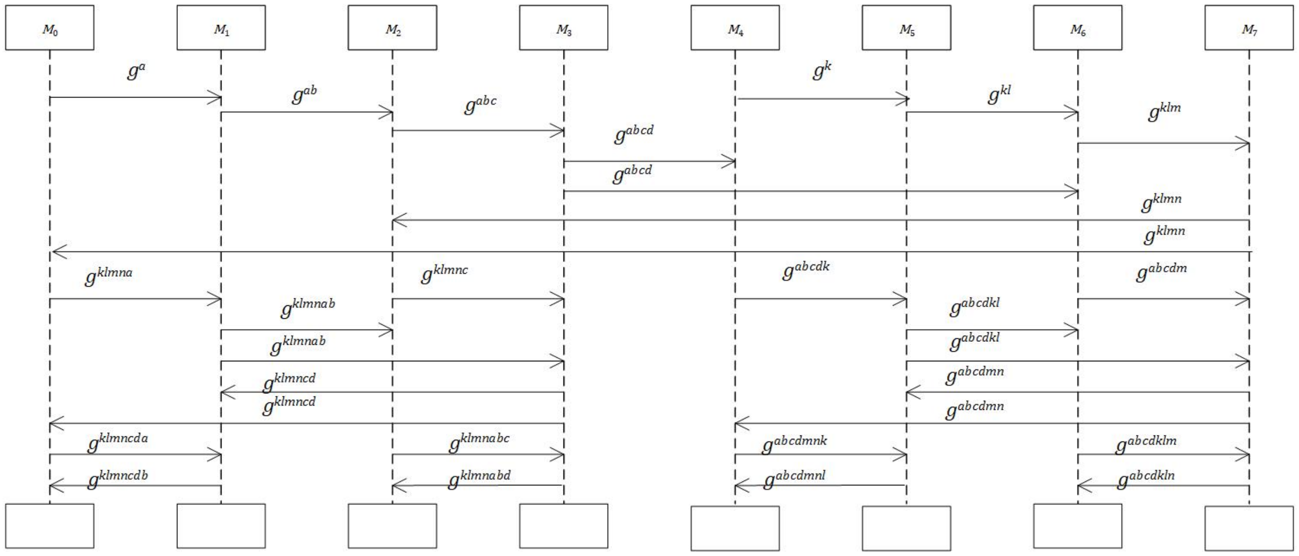

Fig. 2: The message sequence chart for DC-DHKA protocol for eight participants labelled as $M_0, M_1, \ldots, M_7$ whose respective secret exponents are $a, b, c, d, k, \ell, m, n$.

ipants is illustrated in Fig. 2.

The public parameters of the DC-DHKA protocol are identical to those of ING, GDH.1, GDH.2, and GDH.3 protocols. The construction of the mutual secret key involving N participants requires $\lceil \log_2 q \rceil + 1$ phases. Initially, if N is not an exact power of two, then padding is performed. The number of participants after this padding is $2^{\lceil \log_2 N \rceil}$. Notice that if N is already an exact power of two, then $N = 2^{\lceil \log_2 N \rceil}$. Assuming that N is an exact power of two, the first phase of the DC-DHKA protocol consists of two following steps:

1) Initially all participants are labeled with integers from 0 to $N-1$ (inclusive), suppose we denote these participants with $M_0, M_1, \ldots, M_{N-1}$ and their corresponding secret exponents are $s_0, s_1, \ldots, s_{N-1}$. These participants are divided into two subgroups, the first group consists of $M_0, M_2, \ldots, M_{N/2-1}$, whereas the second group consists of $M_{N/2}, M_{N/2+1}, \ldots, M_{N-1}$. In the first group, M_0 starts the first phase by computing g^{s_0} and sends this value M_1, while in parallel $M_{N/2}$ computes $g^{s_{N/2}}$ and sends this value to $M_{N/2+1}$.
2) The second step involves every participant M_i where $1 \leq i \leq N/2-1$ or $N/2+1 \leq i \leq N-1$. Here, M_i receives a value from M_{i-1} and raises it to the power of s_i. Subsequently, one of the following procedures is performed:
   a) if $1 \leq i \leq N/2-2$ or $N/2+1 \leq i \leq N-2$, i.e., M_i is not the last participant in its subgroup, then M_i sends the value from its calculation to M_{i+1};

b) if $i = N/2 - 1$, then $M_{N/2-1}$ transmits the result of its calculation to $M_{N/2}$ and $M_{3N/4}$ (in other words, the last participant of the first subgroup transmits a message to two participants of the second subgroup, namely $M_{N/2}$ and $M_{3N/4}$);
c) if $i = N - 1$, then M_{N-1} transmits the result of its calculation to M_0 and $M_{N/4}$ (that is, the last participant of the second subgroup transmits a message to two participants of the first subgroup, namely M_0 and $M_{N/4}$).

A the end of the first phase, DC-DHKA yields two groups of equal size. The first group consist of $M_0, M_1, \ldots, M_{N/2-1}$, whereas the second one comprises $M_{N/2}, M_{N/2+1}, \ldots, M_{N-1}$. Each group then concurrently performs an analogous phase as described in the first phase of DC-DHKA. Theoretically, this procedure is repeated until the size of the group is 4, which is the base case of the DC-DHKA (see [11], [12]).

B. The Number of Bits Involved in the Transmissions of DC-DHKA Protocol

We are now ready to perform an elementary analysis of the communication complexity of the DC-DHKA protocol. As described in Section III-A, we assume that every message of the form g^x where x is a positive integer requires $\lceil \log_2 q \rceil$ bits of data representation. This convention allows us to compare the number of bits required to construct the mutual secret key in DC-DHKA with the aforementioned results in Section III-C, Section III-D, and Section III-E. Notice that we typically assume that the number of participants in DC-DHKA (denoted by N) is an exact power of two. We first discuss the following lemma.

Lemma 1. *If N is an exact power of two, then the total number of transmissions involving values of the form g^x where x is a positive integer can be expressed as a function $T(N)$ defined recursively as*

$$T(N) = 2T\left(\frac{N}{2}\right) + N + 2, \text{ with base case } T(4) = 10. \quad (9)$$

Notice that we consider two transmissions of an identical message as separate transmissions if at least one of the sender or receiver is different from one transmission to another. For instance, in Fig. 1, the transmissions of g^{ab} from M_1 to M_2 and M_3 are defined as two distinct transmissions. By counting the number of transmissions in the message sequence chart of DC-DHKA for four participants in Fig. 1 we have $T(4) = 10$. Moreover, from the message sequence chart of DC-DHKA for eight participants in Fig. 2 we have $T(8) = 30$. Notice that $T(8)$ can also be obtained from the formula $T(8) = 2T(4) + 8 + 2$. The proof of Lemma 1 is as follows.

Proof of Lemma 1: Assume that initially we have N participants and N is an exact power of two. As in the protocol description, these N participants are divided into two subgroups at the end of the first phase, each group consists of $N/2$ members. The first group comprises $M_0, M_1, \ldots, M_{N/2-1}$, whereas the second group comprises $M_{N/2}, M_{N/2+1}, \ldots, M_{N-1}$. Before the first phase is concluded, each participant M_i where $i = 0, 1, \ldots, N/2 - 1$ or $i = N/2, N/2 + 1, \ldots, N - 2$ transmits one message of the form g^x where x is a positive integer to M_{i+1}. Thus, there are $2 \cdot (N/2 - 1) = N - 2$ transmissions of this category.

For the last participant in each subgroup, observe that $M_{N/2-1}$ sends a message of the form g^x for some positive integer x to $M_{N/2}$ and $M_{3N/4}$, whereas M_{N-1} sends a message of the same form to M_0 and $M_{N/4}$. Hence, there are four additional transmissions for the first phase. By combining this result with the previous one, we obtain $N - 2 + 4 = N + 2$ transmissions that occur in the first phase.

Notice that $T(N/2)$ represents the number of transmissions occurring in a group of $N/2$ participants. Since the groups containing $M_0, M_1, \ldots, M_{N/2-1}$ and $M_{N/2}, M_{N/2+1}, \ldots, M_{N-1}$ are disjoint and independent (there is no communication between any two participants from two disjoint groups after the first phase is finished), then the total number of transmissions in the entire protocol is equal to $N + 2$ plus the total number of transmissions occurring in two subgroups of size $N/2$. Therefore, we conclude that $T(N) = 2T(N/2) + N + 2$. ■

The solution of the recursive function (9) is discussed in the following theorem.

Theorem 1. *If N is an exact power of two, then the solution (the closed form) of the recursive function (9) is*

$$T(N) = N(\log_2 N + 1) - 2. \quad (10)$$

Proof: We prove Theorem 1 using induction on N. If $N = 4$, the statement is true because $T(4) = 10$ from the message sequence chart given in Fig. 1 and $4(\log_2 4 + 1) - 2 = 10$.

Now suppose the statement holds for some $N = 2^K$ where $K \in \mathbb{N}$, that is, $T(N) = T(2^K) = 2^K(\log_2 2^K + 1) - 2 = 2^K(K + 1) - 2$. We need to show that the statement also holds if the number of participants is doubled (the next power of two after N is $2N$). Using (9) and the aforesaid induction hypothesis, we have

$$\begin{aligned} T(2N) &= 2T(N) + 2N + 2 \text{ (using (9))} \\ &= 2T(2^K) + 2 \cdot 2^K + 2 \ (N = 2^K \text{ where } K \in \mathbb{N}) \\ &= 2(2^K(K + 1) - 2) + 2^{K+1} + 2 \text{ (induction hypothesis)} \\ &= 2^{K+1}(K + 1) + 2^{K+1} - 2 \\ &= 2^{K+1}((K + 1) + 1) - 2. \end{aligned}$$

Since $N = 2^K$, then $2^{K+1} = 2N$ and $K + 1 = \log_2 2N$. Thus we obtain $T(2N) = 2N(\log_2 2N + 1) - 2$, i.e., the equation holds if the number of participant is doubled. By mathematical induction, we conclude that (10) holds for any $N \geq 4$ whenever N is an exact power of two. ■

The result in Theorem 1 provides an accurate number of transmissions in DC-DHKA if the number of communicating parties is N and N is an exact power of two. If N is not an exact power of two, then padding must be performed by adding several dummy participants as explained in [11], [12]. Here, the initial number of group members is transformed so that it is an exact power of two. Notice that the nearest exact power of two that is not less than N is $2^{\lceil \log_2 N \rceil}$. As a result, we have the following corollary.

Corollary 1. *Let $T(N)$ be the total number of transmissions in DC-DHKA protocol involving N participants where $N \geq 4$, then*

$$T(N) = 2^{\lceil \log_2 N \rceil} \cdot (\lceil \log_2 N \rceil + 1) - 2. \tag{11}$$

Proof: If N is an exact power of two, then $2^{\lceil \log_2 N \rceil} = N$ and $\lceil \log_2 N \rceil = \log_2 N$ and thus (10) and (11) are identical. Otherwise, the number of participants after the padding is $M = 2^{\lceil \log_2 N \rceil}$ and M is an exact power of two. Based on Theorem 1, we have $T(M) = M \cdot (\log_2 M + 1) - 2$. The result follows from the assumption that the number of participants (including dummies) must be an exact power of two and dummies are also involved in the message transmission. ■

Since we assume that every single transmission in DC-DHKA involves sending a message of the form g^x where x is a positive integer, then each transmission requires a $\lceil \log_2 q \rceil$ bits of data representation if the protocol uses the multiplicative group $\mathbb{F}_q^*$. Accordingly, we obtain the following corollary.

Corollary 2. *Let $t_{DC}(N)$ be the total number of bits required in the entire transmissions of DC-DHKA protocol for N participants that uses the multiplicative group $\mathbb{F}_q^*$, then*

$$t_{DC}(N) = \lceil \log_2 q \rceil \cdot \left[2^{\lceil \log_2 N \rceil} \cdot (\lceil \log_2 N \rceil + 1) - 2\right]. \tag{12}$$

V. Comparison of the Number of Bits Transmitted in Pertinent Protocols

In this section, we compare the of number bits required in the entire communications of GDH.1, GDH.2, GDH.3, and DC-DHKA protocols. We refer to these quantities as the elementary communication complexities of such protocols. Using the result from Section III-B, Section III-C, Section III-D, Section III-E, and Section IV-B, we summarize the elementary communication complexity of each protocol in Table I. For brevity, we refer to GDH.3 version 1 as the variant of GDH.3 protocol that assumes a single transmission of the value $g^{\prod_{k=0}^{N-2} s_k}$ during the first broadcast phase, while GDH.3 version 2 assumes individual transmissions of the same value during the first broadcast phase (which is performed by M_{N-2} to every M_i such that $0 \leq i \leq N-3$).

Assuming that all protocols are defined using an identical finite multiplicative group $\mathbb{F}_q^*$, we notice from Table I that the size of this mathematical structure does not affect the comparison of elementary communication complexity between two different protocols. As a consequence, a comparative analysis of such a complexity between two different protocols can be carried out by determining the number of *elementary transmissions* that occurred in each protocol. Here, an elementary transmission is defined as a transmission of a value in $\mathbb{F}_q^*$, which requires $\lceil \log_2 q \rceil$ bits of data representation.

From Table I, we notice that DC-DHKA protocol asymptotically uses fewer bits than ING, GDH.1, GDH.2 protocols to agree on the mutual secret key. Nevertheless, DC-DHKA protocol asymptotically uses more bits than GDH.3 to achieve the mutual secret key agreement for all communicating parties. Despite this downside, DC-DHKA protocol strictly ensures that all participants obtain the mutual secret key simultaneously and the computational burden (i.e., exponentiation) of each participant is relatively uniform (every member of the group performs one or two exponentiations in each phase, see [11] for details). To provide a more comprehensive comparative analysis among related protocols, we define $t_X(N)$ as a function that returns the number of elementary transmissions that occurred in protocol X with N participants. Thus, we have $t_{ING} = t_{GDH.1} = N(N-1)$, $t_{GDH.2} = (N^2 + 3N - 6)/2$, $t_{GDH.3v1} = 3N - 2$, $t_{GDH.3v2} = 4N - 5$, and $t_{DC} = 2^{\lceil \log_2 N \rceil} \cdot (\lceil \log_2 N \rceil + 1) - 2$. Notice that we override the definition of t_{DC} in (12) by ignoring the term $\lceil \log_2 q \rceil$. For all related protocols, we perform a numerical experiment to compute $t_X(N)$ for all $4 \leq N \leq 100$. The graph for all $t_X(N)$ where $X \in \{ING, GDH.1, GDH.2, GDH.3v1, GDH.3v2, DC\}$ and $4 \leq N \leq 30$ is depicted in Fig. 3.

From the numerical experiment, we obtain that the number of elementary transmissions of DC-DHKA is always fewer than those in ING, GDH.1, and GDH.2 protocols for $N \geq 19$ participants. Moreover, DC-DHKA is always more efficient than ING and GDH.1 protocols in terms of communications if we consider a group of at least ten participants. We infer the following proposition using the experimental data and mathematical induction on the number of communicating parties.

Proposition 1. *Let $t_{ING}(N)$, $t_{GDH.1}(N)$, $t_{GDH.2}(N)$, and $t_{DC}(N)$ respectively denote the number of elementary transmissions occurred in ING, GDH.1, GDH.2, and DC-DHKA protocols for a group of N participants, then*

1) $t_{DC}(N) < t_{ING}(N) = t_{GDH.1}(N)$ *for every* $N \in \{4, 7, 8\}$ *and* $N \geq 10$,
2) $t_{DC}(N) < t_{GDH.2}(N)$ *for every* $N \in \{4, 7, 8\}$, $N \in \{12, \ldots, 16\}$, *and* $N \geq 19$.

From experimental data, we also obtain that $t_{GDH.3v2}(4) > t_{DC}(4)$. Thus $N = 4$ is the only condition in which the DC-DHKA protocol requires fewer bits in the communication than the GDH.3 version 2 protocol.

VI. Concluding Remarks

In Corollary 1, we show that the total number of elementary transmissions in DC-DHKA protocol involving $N \geq 4$ participants is $2^{\lceil \log_2 N \rceil} \cdot (\lceil \log_2 N \rceil + 1) - 2$. Furthermore, assuming that each elementary transmission requires $\lceil \log_2 q \rceil$ bits of data representation, we arrive with a conclusion that the total number of bits needed to be communicated in the entire DC-DHKA protocol is $\lceil \log_2 q \rceil \cdot \left[2^{\lceil \log_2 N \rceil} \cdot (\lceil \log_2 N \rceil + 1) - 2\right]$. This result shows that the DC-DHKA protocol asymptotically uses fewer bits in communication than ING, GDH.1, and GDH.2 protocols to agree on a mutual secret key. By combining this outcome with the previous conclusion regarding the total number of exponentiations involved in ING, GDH.1, and GDH.2 protocols in [11], we infer that DC-DHKA is asymptotically more efficient than the other three protocols in terms of the

TABLE I

The number of bits involved in the entire communications of ING, GDH.1, GDH.2, GDH.3, and DC-DHKA protocols with N participants. All protocols use the multiplicative group $\mathbb{F}_q^*$.

protocol name	number of bits used in the entire transmissions	asymptotic expression in N
ING	$\lceil \log_2 q \rceil \cdot N \cdot (N-1)$	$\mathcal{O}(N^2)$
GDH.1	$\lceil \log_2 q \rceil \cdot N \cdot (N-1)$	$\mathcal{O}(N^2)$
GDH.2	$\frac{1}{2} \cdot \lceil \log_2 q \rceil (N^2 + 3N - 6)$	$\mathcal{O}(N^2)$
GDH.3 version 1	$\lceil \log_2 q \rceil \cdot (3N - 2)$	$\mathcal{O}(N)$
GDH.3 version 2	$\lceil \log_2 q \rceil \cdot (4N - 2)$	$\mathcal{O}(N)$
DC-DHKA	$\lceil \log_2 q \rceil \cdot \left[2^{\lceil \log_2 N \rceil} \cdot (\lceil \log_2 N \rceil + 1) - 2\right]$	$\mathcal{O}(N \log N)$

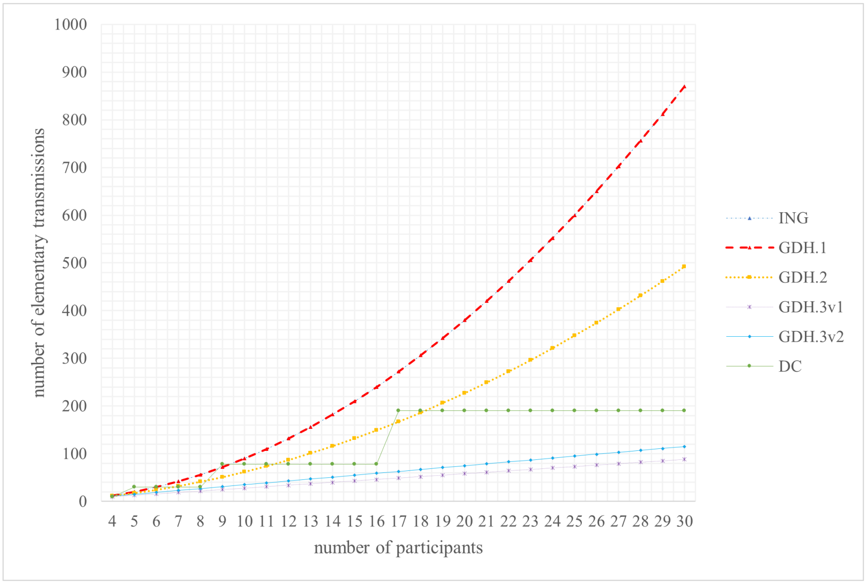

Fig. 3: The number of elementary transmissions in ING, GDH.1, GDH.2, GDH.3 and DC-DHKA protocols. The GDH.3 version 1 protocol (GDH.3v1) assumes single transmission of the value $g^{\prod_{k=0}^{N-2} s_k}$, whereas the GDH.3 version 2 protocol (GDH.3v2) assumes that the same value is transmitted individually by M_{N-2} to all M_i such that $0 \leq i \leq N-3$. Notice that since $t_{ING}(N) = t_{GDH.1}(N)$, then the graphs of both functions are coincide.

number of exponentiations and the number of bits needed to be communicated for constructing the mutual secret key. In addition, by considering the numerical experiments in Section V and [11], we obtain $N = 19$ as the theoretical crossover point at which the DC-DHKA protocol is more efficient than the other three protocols in respect of the two aforesaid criteria.

Our analysis and experiment show that DC-DHKA asymptotically uses more bits in its transmission than the GDH.3 protocol. Combined with the outcome in [11], we infer that GDH.3 is more efficient than DC-DHKA in the matter of the number of exponentiations and bits required in the entire transmission. Moreover, Steiner et al. show that GDH.2 and GDH.3 have efficient procedures for addressing group membership modification (i.e., insertion and removal of members to the group) [8], and such procedures are not available in DC-DHKA. Despite these advantages, formal analysis in [12] shows that the mutual key retrievals in GDH.1, GDH.2, and GDH.3 do not occur strictly all at once for all communicating parties. As a result, the DC-DHKA protocol is preferred to GDH.1, GDH.2, and GDH.3 protocols if we consider a scenario in which every participant must obtain the mutual key strictly simultaneously. Another group key agreement with the regular DH key that obliges its participant to retrieve the mutual key strictly at the same time is the ING protocol. However, as discussed in Section III-B and [11], the ING protocol asymptotically uses more bits in its communication and requires more exponentiations.

Our analysis provides elementary communication complexities for pertinent protocols that use regular DH keys as their mutual secret keys. This complexity measures the number of bits required to be communicated for constructing the group key. We show that two different protocols may have identical elementary communication complexity even though the number of transmissions in each protocol is different. For example, GDH.1 involves fewer transmissions than the ING protocol (regarding the number of messages sent and received as well as the number of communications), but the total number of bits required for creating the mutual secret key in both protocols are identical. This type of analysis can also be performed to investigate the communication complexities of non-Diffie-Hellman-based group key agreements, such as matrix-vector-based group key agreements (such as in [19]) or post-quantum key agreement protocol (such as in [20]).

References

[1] T. Hardjono and L. R. Dondeti, *Security in Wireless LANS and MANS (Artech House Computer Security)*. Artech House, Inc., 2005.

[2] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," in *2017 international conference on engineering and technology (ICET)*. IEEE, 2017, pp. 1–7.

[3] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 1, pp. 60–96, 2004.

[4] 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.

[5] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. IT.22 No. 6, pp. 644–654, 1976.

[6] 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.

[7] 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.

[8] 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.

[9] K. Becker and U. Wille, "Communication complexity of group key distribution," in *Proceedings of the 5th ACM conference on Computer and communications security*, 1998, pp. 1–6.

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

[11] R. Dewoprabowo, M. Arzaki, and Y. Rusmawati, "On Generalized Divide and Conquer Approach for Group Key Distribution: Correctness and Complexity," in *2018 6th International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2018, pp. 416–424.

[12] ——, "Formal Verification of Divide and Conquer Key Distribution Protocol Using ProVerif and TLA+," in *2018 10th International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE, 2018, pp. 451–458.

[13] A. Rao and A. Yehudayoff, *Communication Complexity: and Applications*. Cambridge University Press, 2020.

[14] A. C.-C. Yao, "Some complexity questions related to distributive computing (preliminary report)," in *Proceedings of the eleventh annual ACM symposium on Theory of computing*, 1979, pp. 209–213.

[15] E. Kushilevitz, "Communication complexity," in *Advances in Computers*. Elsevier, 1997, vol. 44, pp. 331–360.

[16] I. Haitner, N. Mazor, R. Oshman, O. Reingold, and A. Yehudayoff, "On the communication complexity of key-agreement protocols," *arXiv preprint arXiv:2105.01958*, 2021.

[17] J. Hoffstein, J. C. Pipher, and J. H. Silverman, *An introduction to mathematical cryptography*, 2nd ed. Springer, 2014.

[18] K. H. Rosen, *Discrete Mathematics and Its Applications, 8th Edition*. McGraw-Hill Education, 2018. [Online]. Available: https://books.google.co.id/books?id=rwZLAgAAQBAJ

[19] 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.

[20] X. L. Jintai Ding, Xiang Xie, "A simple provably secure key exchange scheme based on the learning with errors problem," Cryptology ePrint Archive, Report 2012/688, 2012, https://eprint.iacr.org/2012/688.