OPEN ACCESS
Indo-JC
Indonesian Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 2, Issue. 2, Sept 2017. pp. 55-78
doi:10.21108/indojc.2017.22.179

On the Generalizations of Megrelishvili Protocol for Group Key Distribution

Muhammad Arzaki

Computing Laboratory, School of Computing, Telkom University
Bandung (40257) Indonesia

arzaki@telkomuniversity.ac.id

Abstract

This article presents an extension of our previous research in [1] where we propose two variants of Megrelishvili key distribution schemes and investigate some of their elementary theoretical security analysis. We briefly discuss the two protocols in [1] and propose another two schemes which are more efficient than the preceding ones. Additionally, we also devise effective procedures for constructing a new mutual key if the group membership is altered. Furthermore, we discuss the security of the protocols rigorously and we provide a sufficient condition for breaking the protocols by way of solving several instances of *Megrelishvili vector-matrix problems* (MVMP). We prove that the secret group key can be recovered easily if an attacker can express the sum of the secret exponents of the participants as a linear combination of the secret exponents excerpted from the transmission. Based on this result, we reason that our Megrelishvili key distribution schemes are theoretically at least as secure as the standard two-party Megrelishvili key exchange procedure.

Keywords: Megrelishvili protocol, Megrelishvili key distribution scheme, key distribution scheme, protocol extension

Abstrak

Artikel ini menyajikan sebuah pengembangan dari penelitian kami sebelumnya di [1] yang berisi pengajuan dua varian skema distribusi kunci Megrelishvili dan pembahasan analisis teori keamanan dasar mereka. Kami membahas secara singkat dua protokol di [1] dan mengajukan dua skema lain yang lebih efisien daripada yang sebelumnya. Selain itu, kami juga merancang prosedur yang efektif untuk membangun kunci bersama yang baru jika keanggotaan kelompok diubah. Selanjutnya, kami membahas keamanan protokol-protokol secara sistematis dan menyajikan suatu kondisi cukup (*sufficient condition*) untuk memecahkan protokol-protokol tersebut dengan cara memecahkan beberapa kasus masalah vektor-matriks Megrelishvili (*Megrelishvili vector-matrix problem*, MVMP). Kami membuktikan bahwa kunci rahasia suatu kelompok dapat ditemukan dengan mudah bila penyerang dapat mengekspresikan jumlah pangkat rahasia dari seluruh peserta sebagai kombinasi linier dari pangkat rahasia yang diekstraksi dari transmisi. Berdasarkan hasil ini, kami beranggapan bahwa skema distribusi kunci Megrelishvili secara teori setidaknya sama amannya dengan prosedur pertukaran kunci Megrelishvili standar dua pihak.

Kata Kunci: protokol Megrelishvili, skema distribusi kunci Megrelishvili, skema distribusi kunci, pengembangan protokol

I. Introduction

The group communication among multiple participants is one of the omnipresent occurrence in our modern world. On many occasions, this communication needs to be protected from unauthorized parties. The protection can be achieved in many ways, one of them is by the creation of a mutual secret key among legitimate participants. The efficient construction of the mutual secret key for group communication

Received on Aug 2017. Accepted on Sept 2017

has been one of the central discussions in modern cryptography. Since the development of the Diffie-Hellman key exchange (DHKE) procedure in [2], numerous protocols for the group key creation have been proposed [3]–[6]. Despite these abundant options, the security of these key generation algorithms depends on somewhat equivalent discrete logarithm problems in finite groups—which have been extensively investigated in numerous studies [7]–[10].

The original concept of Megrelishvili key exchange procedure was first discussed in [11]. This protocol can be considered as a linear algebra-based variant of the DHKE. It combines vector-matrix multiplication and matrix exponentiation for the construction of the mutual secret key. These linear algebraic operations cause the security of Megrelishvili protocol does not directly relate to a specific discrete logarithm problem in a finite group. Thus, Megrelishvili key exchange procedure offers an alternative method for the construction of a mutual key between two parties.

The objective of this article is to present an extension of our previous work in [1]—where we propose two variants of multi-party Megrelishvili protocols and discuss their elementary theoretical security analysis. These multi-party Megrelishvili protocols allow multiple participants to establish a common mutual key using the computational characteristics of the standard two-party Megrelishvili key exchange. In this article, we briefly discuss the two protocols in [1] and we propose another two key distribution schemes which are more efficient than the previous protocols. Our construction idea of these protocols is based on the extension idea of the generic two-party DHKE to multiple participants described in [3], [5]. The correctness and some important characteristics of our key distribution schemes are discussed rigorously. In addition, we also devise membership modification protocols for two of our schemes. These additional protocols allow the group to add a new or delete an existing participant after the initial key distribution is completed without running a complete re-execution of the procedure.

This article also presents a rigorous elementary theoretical security analysis of our Megrelishvili key distribution schemes. The analysis is focused on the investigation of *multi-party Megrelishvili shared key problem* (MMSKP) which was first addressed in [1]. We prove that MMSKP can be solved by means of finding the solution of several MVMP (*Megrelishvili vector-matrix problem*) instances. Furthermore, we show that the mutual secret key of the group can be computed efficiently if an attacker can express the sum of the secret exponents of the participants as a linear combination of the secret exponents excerpted from the transmission. By this result, we reason that our Megrelishvili key distribution procedures are at least as secure as the original two-party Megrelishvili key exchange.

The rest of this article is organized as follows. We present our linear algebra notations and terminologies in Section II. Next, we discuss some of our related works, e.g., the two-party and three-party Megrelishvili protocols and some of their characteristics in Section III. The generalizations of Megrelishvili protocol for key distribution scheme and their analyses are explained in Section IV. Section V discusses our proposed membership modification protocols for two types of our key distribution procedure. The security of the protocols are then analyzed in Section VI. Finally, some of the important concluding remarks are summarized in Section VII.

II. Linear Algebra Notations and Terminologies

Before we discuss some variants of Megrelishvili key distribution and their properties rigorously, we first explain some of our notations and terminologies in linear algebra over finite fields. We use similar notations as in [1], [12]–[14]. Throughout this paper, $\mathbb{F}_q$ denotes the finite field of q elements and $\mathbb{F}_q^n$ denotes the n-dimensional vector space over $\mathbb{F}_q$. We use boldface lowercase letters (e.g., $\mathbf{v}$) to denote all vectors in $\mathbb{F}_q^n$ and boldface uppercase letters (e.g., $\mathbf{M}$) to denote all matrices over $\mathbb{F}_q$. The vectors are mostly considered and handled as row matrices, unless otherwise specified (e.g., in Section VI). Therefore, for any $\mathbf{v} \in \mathbb{F}_q^n$ and any $n \times n$ matrix $\mathbf{M}$ over $\mathbb{F}_q$, the expression $\mathbf{vM}$ is well-defined and is a vector in $\mathbb{F}_q^n$. Moreover, the expression $\mathbf{vM}$ is called a left-multiplication of $\mathbf{M}$ by $\mathbf{v}$, or equivalently, a right-multiplication of $\mathbf{v}$ by $\mathbf{M}$. The *order* of a nonsingular matrix $\mathbf{M}$ is the smallest positive integer s that makes $\mathbf{M}^s = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. According to [15], the order of any $n \times n$ nonsingular matrix over $\mathbb{F}_q$ is always less than or equal to $q^n - 1$. Given N invertible matrices $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_N$, we define $\prod_{k=1}^{N} \mathbf{M}_k = \mathbf{M}_1 \mathbf{M}_2 \cdots \mathbf{M}_n$ and $\prod_{k=i}^{j} \mathbf{M}_k = \mathbf{M}_i \mathbf{M}_{i+1} \cdots \mathbf{M}_{j-1} \mathbf{M}_j$ if $1 \leq i \leq j \leq N$.

For convenience, we define the empty product of invertible matrix, that is $\prod_{k=i}^{j} \mathbf{M}_k$ with $i > j$, as the identity matrix.

Although for most part of the paper the matrices and vector spaces are considered over a finite field, we utilize elementary linear algebra over real field to discuss some theoretical security aspects of our protocols in Section VI. In this case, the vectors are typically considered and treated as column matrices. Thus, for any $\mathbf{v} \in \mathbb{R}^n$ and any $m \times n$ matrix $\mathbf{M}$ over $\mathbb{R}$, the expression $\mathbf{Mv}$ is well-defined and is a vector in $\mathbb{R}^m$.

III. Some Related Works

A. Two-Party Megrelishvili Protocol, MVMP, and MSKP

The two-party Megrelishvili protocol is an example of linear algebra-based variant of the Diffie-Hellman key exchange. The theoretical concept of this protocol was first discussed in [11] by R. Megrelishvili, M. Chelidze, and K. Chelidze. Since then, the study concerning this protocol has been conducted by numerous researchers (e.g.: [1], [12]–[14], [16]–[19]). In this section, we briefly discuss the formal description of the generic two-party protocol using the notations used in [1], [12]–[14], [19]. The reader is referred to [12] for the discussion concerning the algorithm analyses and the comparison of the standard two-party Megrelishvili protocol to other prominent variants of the Diffie-Hellman key agreement.

The mutual secret key in Megrelishvili protocol is a vector in $\mathbb{F}_q^n$. Before the key exchange between two participants takes place, a trusted third party chooses and publishes several public parameters, i.e.: a finite field $\mathbb{F}_q$, a nonsingular matrix $\mathbf{M}$ of size $n \times n$ over $\mathbb{F}_q$, and a nonzero vector $\mathbf{v} \in \mathbb{F}_q^n$. The matrix $\mathbf{M}$ is chosen in such a way that its order is sufficiently large. One method to construct the public matrix $\mathbf{M}$ and the public nonzero vector $\mathbf{v}$ is discussed in [13]. The author in [13] suggested to choose $\mathbf{M}$ that is *similar* to a companion matrix of a primitive polynomial over $\mathbb{F}_q$[1]. By using such $\mathbf{M}$, Theorem 3 in [13] ensures that the value of $\mathbf{vM}^t$ are all nonzero and distinct for any nonzero vector $\mathbf{v} \in \mathbb{F}_q^n$ and $t \in [0, q^n - 2]$. The construction method for $\mathbf{M}$ and $\mathbf{v}$ in [13] also provides a maximal possible key space in Megrelishvili protocol, i.e., any nonzero vector $\mathbf{w} \in \mathbb{F}_q^n$ can be expressed as $\mathbf{vM}^t$ for some integer t.

Suppose the participants in the two-party Megrelishvili protocol are participant 1 (P_1) and participant 2 (P_2). To generate the mutual secret vector, each participant i picks a secret integer α_i and constructs a private matrix $\mathbf{P}_i = \mathbf{M}^{\alpha_i}$. Afterward, each participant i computes the vector $\mathbf{a}_i = \mathbf{vP}_i$ and transmits this value to another participant in an open channel. To retrieve the mutual vector, each participant right-multiplies the received vector by its own private matrix. Because $\mathbf{P}_1\mathbf{P}_2 = \mathbf{M}^{\alpha_1}\mathbf{M}^{\alpha_2} = \mathbf{M}^{\alpha_1+\alpha_2} = \mathbf{M}^{\alpha_1}\mathbf{M}^{\alpha_2} = \mathbf{P}_2\mathbf{P}_1$, we have $\mathbf{vP}_1\mathbf{P}_2$= $\mathbf{vP}_2\mathbf{P}_1$, and thus the key exchange is completed. This protocol is summarized in Table I.

Since the exchange of vectors is performed over an open channel, the value of $\mathbf{a}_i = \mathbf{vP}_i = \mathbf{vM}^{t_i}$ is publicly known. This condition makes an attacker knows the values of $\mathbf{a}_1$ and $\mathbf{a}_2$. By observation, the attacker can acquire the mutual secret vector by solving the equation $\mathbf{a}_i = \mathbf{vM}^{t_i}$ for t_i, where $i \in \{1, 2\}$, and then computing the value of $\mathbf{vM}^{t_1+t_2}$ in a polynomial number of scalar operations in $\mathbb{F}_q$. Hence, from mathematical perspective, the security of Megrelishvili protocol strongly relates to the *Megrelishvili vector-matrix problem* (MVMP), that is, the problem of determining the value t from the equation $\mathbf{vM}^t = \mathbf{w}$ where $\mathbf{M}$ is a nonsingular matrix and both $\mathbf{v}$ and $\mathbf{w}$ are vectors of compatible size [1], [12]–[14]. However, the actual objective of the attacker is to solve the *Megrelishvili shared key problem* (MSKP), that is, the problem of computing the vector $\mathbf{vM}^{t_1+t_2}$ from the known values of $\mathbf{vM}^{t_1}$ and $\mathbf{vM}^{t_2}$ [1], [14]. It is evident that the attacker can solve the MSKP by solving the MVMP first. Moreover, if the attacker has retrieved the value of t_1 and t_2, then the value $\mathbf{vM}^{t_1+t_2}$ can be computed in a polynomial number of scalar operations in $\mathbb{F}_q$. This condition also implies that the MSKP is not computationally harder than the MVMP. Nevertheless, to our knowledge, the formal relationship between MVMP and MSKP has not been comprehensively explored.

[1]That is, $\mathbf{M} = \mathbf{P}^{-1}\mathbf{CP}$ for some invertible matrix $\mathbf{P}$ and a matrix $\mathbf{C}$ whose characteristic polynomial is a primitive polynomial over $\mathbb{F}_q$.

Setup for public parameters	
A trusted third party announce: a finite field $\mathbb{F}_q$, a (large) integer n, an $n \times n$ nonsingular matrix $\mathbf{M}$ over $\mathbb{F}_q$, and a nonzero vector $\mathbf{v} \in \mathbb{F}_q^n$.	
Generation of the private matrices	
Participant 1 (P_1)	*Participant 2* (P_2)
Pick an integer α_1. Calculate $\mathbf{P}_1 = \mathbf{M}^{\alpha_1}$.	Pick an integer α_2. Calculate $\mathbf{P}_2 = \mathbf{M}^{\alpha_2}$.
Exchange of vectors via an open channel	
Participant 1 (P_1)	*Participant 2* (P_2)
Compute $\mathbf{a}_1 = \mathbf{v}\mathbf{P}_1$. P_1 transmits $\mathbf{a}_1$ to P_2.	Compute $\mathbf{a}_2 = \mathbf{v}\mathbf{P}_2$. P_2 transmits $\mathbf{a}_2$ to P_1.
Mutual secret key retrieval	
Participant 1 (P_1)	*Participant 2* (P_2)
Compute $\mathbf{a}_1' = \mathbf{a}_2\mathbf{P}_1$.	Compute $\mathbf{a}_2' = \mathbf{a}_1\mathbf{P}_2$.
The common secret vector is $\mathbf{a}_1' = \mathbf{a}_2'$.	

TABLE I: Two-party Megrelishvili protocol according to [1], [12]–[14], [19].

One straightforward method to solve the MVMP is by using the brute-force (exhaustive search) attack described in [12]–[14]. If the order of the public matrix $\mathbf{M}$ is known, the attacker simply computes the sequence of vectors $\mathbf{v}\mathbf{M}^0$, $\mathbf{v}\mathbf{M}^1$, $\ldots$, $\mathbf{v}\mathbf{M}^b$, where b is the order of $\mathbf{M}$. However, if the order of the public matrix is unknown, the attacker can assume that the order of $\mathbf{M}$ is maximal, i.e., $b = q^n - 1$. The brute-force method can solve the MVMP in $\mathbb{F}_q^n$ using $\mathcal{O}\left(n^3 \cdot q^n\right)$ scalar operations under the assumption that the standard $\mathcal{O}\left(n^3\right)$ matrix multiplication algorithm is used for exponentiating the matrix and the order of the public matrix is bounded by $q^n - 1$. Besides this straightforward approach, there exists a non-trivial method for solving the MVMP using collision algorithm described in [14]. Under the similar aforementioned assumption, this algorithm requires at most $\mathcal{O}\left(\log q \cdot n^4 \cdot q^{n/2}\right)$ scalar operations for solving the MVMP in $\mathbb{F}_q^n$, which means that it is faster than the exhaustive search attack by a factor of $\mathcal{O}\left(\left(1/n \log q\right) \cdot q^{n/2}\right)$. Nevertheless, the collision algorithm requires more storage during its execution [14].

B. Three-Party Megrelishvili Protocol

In this section, we explain the extension of the two-party Megrelishvili protocol to a group containing three members, namely P_0, P_1, and P_2[2]. This relatively straightforward extension was first discussed [1] and it uses identical public parameters as in the two-party key exchange. Like the two-party protocol, each member P_i initially generates a private matrix $\mathbf{P}_i$ by choosing an integer α_i and setting $\mathbf{P}_i = \mathbf{M}^{\alpha_i}$.

The key exchange for three members consists of two rounds (or stages) of transmissions[3]. We define $\mathbf{a}_i^j$ as a vector computed by P_i at round j. At the beginning, each P_i calculates $\mathbf{a}_i^0 = \mathbf{v}\mathbf{P}_i$. In the first round, each P_i sends $\mathbf{a}_i^0$ to $P_{(i+1) \bmod 3}$. This means P_0 sends $\mathbf{a}_0^0$ to P_1, P_1 sends $\mathbf{a}_1^0$ to P_2, and P_2 sends $\mathbf{a}_2^0$ to P_1. Each member P_i then right-multiplies the vector received by its own private matrix, the resulting vector is denoted by $\mathbf{a}_i^1$. Notice that we have $\mathbf{a}_0^1 = \mathbf{a}_2^0\mathbf{P}_0 = \mathbf{v}\mathbf{P}_2\mathbf{P}_0$, $\mathbf{a}_1^1 = \mathbf{a}_0^0\mathbf{P}_1 = \mathbf{v}\mathbf{P}_0\mathbf{P}_1$, and $\mathbf{a}_2^1 = \mathbf{a}_1^0\mathbf{P}_2 = \mathbf{v}\mathbf{P}_1\mathbf{P}_2$. In the second round, each P_i sends $\mathbf{a}_i^1$ to the identical recipient as in the first round. To retrieve the mutual key, each member simply right-multiplies the vector received by its own private matrix. At this point, we have $\mathbf{a}_0^2 = \mathbf{a}_1^2 = \mathbf{a}_2^2$ because

$$\mathbf{a}_0^2 = \mathbf{a}_2^1\mathbf{P}_0 = \mathbf{v}\mathbf{P}_1\mathbf{P}_2\mathbf{P}_0, \tag{1}$$

$$\mathbf{a}_1^2 = \mathbf{a}_0^1\mathbf{P}_1 = \mathbf{v}\mathbf{P}_2\mathbf{P}_0\mathbf{P}_1, \tag{2}$$

$$\mathbf{a}_2^2 = \mathbf{a}_1^1\mathbf{P}_2 = \mathbf{v}\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2, \tag{3}$$

and $\mathbf{P}_1\mathbf{P}_2\mathbf{P}_0 = \mathbf{P}_2\mathbf{P}_0\mathbf{P}_1 = \mathbf{P}_0\mathbf{P}_1\mathbf{P}_2 = \mathbf{M}^{\alpha_0+\alpha_1+\alpha_2}$. This protocol is summarized in Table II. We refer the reader to [1, Example 1] for small computational example of this protocol.

[2]To ease our subsequent analysis, we start the index of the group member with 0.

[3]One round or one stage is loosely defined as one series of transmission that involves all members.

Setup for public parameters		
The public parameters are identical to those described for the two-party protocol.		
Generation of the private matrices		
Participant 0 (P_0)	**Participant 1** (P_1)	**Participant 2** (P_2)
Pick an integer α_0. Compute $\mathbf{P}_0 = \mathbf{M}^{\alpha_0}$.	Pick an integer α_1. Compute $\mathbf{P}_1 = \mathbf{M}^{\alpha_1}$.	Pick an integer α_2. Compute $\mathbf{P}_2 = \mathbf{M}^{\alpha_2}$.
The values α_0, α_1, α_2, $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$ are private.		
Initialization (***round*** **0**)		
Participant 0 (P_0)	**Participant 1** (P_1)	**Participant 2** (P_2)
Compute $\mathbf{a}_0^0 = \mathbf{v}\mathbf{P}_0$.	Compute $\mathbf{a}_1^0 = \mathbf{v}\mathbf{P}_1$.	Compute $\mathbf{a}_2^0 = \mathbf{v}\mathbf{P}_2$.
Round **1**		
Public exchange of vectors		
P_0 sends $\mathbf{a}_0^0$ to P_1.	P_1 sends $\mathbf{a}_1^0$ to P_2.	P_2 sends $\mathbf{a}_2^0$ to P_0.
Private computation of vectors		
P_0 computes $\mathbf{a}_0^1 = \mathbf{a}_2^0\mathbf{P}_0$.	P_1 computes $\mathbf{a}_1^1 = \mathbf{a}_0^0\mathbf{P}_1$.	P_2 computes $\mathbf{a}_2^1 = \mathbf{a}_1^0\mathbf{P}_2$.
Round **2**		
Public exchange of vectors		
P_0 sends $\mathbf{a}_0^1$ to P_1.	P_1 sends $\mathbf{a}_1^1$ to P_2.	P_2 sends $\mathbf{a}_2^1$ to P_0.
Private computation of vectors		
P_0 computes $\mathbf{a}_0^2 = \mathbf{a}_2^1\mathbf{P}_0$.	P_1 computes $\mathbf{a}_1^2 = \mathbf{a}_0^1\mathbf{P}_1$.	P_2 computes $\mathbf{a}_2^2 = \mathbf{a}_1^1\mathbf{P}_2$.
The mutual secret key is $\mathbf{a}_0^2 = \mathbf{a}_1^2 = \mathbf{a}_2^2$.		

TABLE II: Megrelishvili key distribution for three members as explained in [1].

IV. Some Generalizations of Megrelishvili Protocol for Group Key Distribution

This section discusses four different generalizations of Megrelishvili protocol for a group of N participants and their elementary characteristics. The first two schemes have also been discussed in [1]. Our idea is based on the extension of the generic two-party Diffie-Hellman key exchange to group communication explained in [5].

A. *Generic Multi-Party Megrelishvili Key Distribution*

We first discuss a straightforward generalization of the three-party Megrelishvili protocol previously explained in Section III-B to a group of N members. This procedure was first described in [1] and it uses the same public parameters as in the two-party and the three-party protocols. Suppose the members of the group are $P_0, P_1, \ldots, P_{N-1}$. At the beginning, each member P_i selects a secret integer α_i and computes the private matrix $\mathbf{P}_i = \mathbf{M}^{\alpha_i}$. If $\mathbf{M}$ is an $n \times n$ matrix over $\mathbb{F}_q$ of maximal order, the value of α_i can be drawn randomly from the integers in $[0, q^n - 2]$. One objective of the protocol is to ensure that each member eventually obtains the mutual vector $\mathbf{v}\mathbf{M}^s$ where $s = \sum_{i=0}^{N-1} \alpha_i$.

To establish the mutual secret key, initially the members are arranged in a circular configuration as in [3]. With this configuration, each participant P_i always transmits its messages to participant $P_{(i+1) \bmod N}$. The protocol uses $N-1$ rounds of vectors transmission to establish the mutual secret key. To facilitate our analysis, let $\mathbf{a}_i^j$ be a vector computed by P_i at round j where $0 \leq i, j \leq N-1$. Before the key exchange occurs, each member P_i calculates $\mathbf{a}_i^0 = \mathbf{v}\mathbf{P}_i$. In the first round, P_i transmits $\mathbf{a}_i^0$ to $P_{(i+1) \bmod N}$, and subsequently computes $\mathbf{a}_i^1 = \mathbf{a}_{(i-1) \bmod N}^0 \mathbf{P}_i$. In general, at round j where $1 \leq j \leq N-1$, participant P_i sends $\mathbf{a}_i^{j-1}$ to $\mathbf{P}_{(i+1) \bmod N}$ and calculates $\mathbf{a}_i^j = \mathbf{a}_{(i-1) \bmod N}^{j-1} \mathbf{P}_i$.

The uniformity of the mutual secret vectors in this protocol originates from the commutativity of the private matrices' product. This means the product $\prod_{i \in [0,N-1]} \mathbf{P}_i$ is the same regardless the order of the matrices. Using mathematical induction, we can prove that $\mathbf{a}_i^j = \mathbf{a}_{(i-1) \bmod N}^{j-1} \mathbf{P}_i = \mathbf{v} \prod_{k=0}^{j} \mathbf{P}_{(i-j+k) \bmod N}$

for $0 \leq j \leq N-1$. Consequently, we get

$$\mathbf{a}_i^{N-1} = \mathbf{v} \prod_{k=0}^{N-1} \mathbf{P}_{(i-N+1+k) \bmod N} \tag{4}$$

$$= \mathbf{v}\mathbf{P}_{(i-N+1) \bmod N}\mathbf{P}_{(i-N+2) \bmod N} \cdots \mathbf{P}_{(i-1) \bmod N}\mathbf{P}_i = \mathbf{v} \prod_{k=0}^{N-1} \mathbf{P}_k, \tag{5}$$

for all $0 \leq i \leq N-1$. The last equality in (5) ensures that each participant gets identical vector after $N-1$ rounds of messages transmission. The reader may refer to [1, Theorem 1] for more detailed explanation regarding the correctness proof of this protocol.

We summarize the protocol in Table III and present its simulation procedure in Algorithm 1. By observation, each member in the protocol only performs one matrix exponentiation for computing the private matrix during the initialization step. Hence, this generic multi-party Megrelishvili protocol differs from the conventional multi-party DHKE in [3] where exponentiation is always performed in each of the rounds. The most extensively used operation in this protocol is the right-multiplication (i.e., the vector-matrix multiplication), which can be performed using $\mathcal{O}\left(n^2\right)$ scalar operations in $\mathbb{F}_q$.

Setup for public parameters
The public parameters are identical to those described for the two-party protocol.
Generation of the private matrices
For all $0 \leq i \leq N-1$, each participant P_i selects a random integer α_i and generates the private matrix $\mathbf{P}_i = \mathbf{M}^{\alpha_i}$. The values α_i and $\mathbf{P}_i$ are both private.
Initialization (round 0)
For all $0 \leq i \leq N-1$, each participant P_i computes $\mathbf{a}_i^0 = \mathbf{v}\mathbf{P}_i$.
Round j, where $1 \leq j \leq N-1$
P_i sends $\mathbf{a}_i^{j-1}$ to $P_{(i+1) \bmod N}$. P_i calculates $\mathbf{a}_i^j = \mathbf{a}_{(i-1) \bmod N}^{j-1}\mathbf{P}_i$
The shared key is $\mathbf{a}_i^{N-1}$ for every $0 \leq i \leq N-1$.

TABLE III: A straightforward extension of Megrelishvili protocol for N participants as described in [1].

Algorithm 1 A procedure for simulating the generic multi-party Megrelishvili protocol as in [1].

```
Require: Public parameters as explained in Table III and an integer $N \geq 2$ which denotes the number
    of group members.
 1: for $i \leftarrow 0$ to $N-1$ do                                   // $N$ group members
 2:     $\mathbf{P}_i \leftarrow \mathbf{M}^{\alpha_i}$                 // private matrices generation for $N$ group members
 3: end for
 4: for $i \leftarrow 0$ to $N-1$ do                                   // $N$ group members
 5:     $\mathbf{a}_i^0 \leftarrow \mathbf{v}\mathbf{P}_i$             // initialization of $\mathbf{a}_i^0$ for $0 \leq i \leq N-1$
 6: end for
 7: for $j \leftarrow 1$ to $N-1$ do                                   // $N-1$ rounds of transmission
 8:     for $i \leftarrow 0$ to $N-1$ do                               // $N$ group members
 9:         $P_i$ sends $\mathbf{a}_i^{j-1}$ to $P_{(i+1) \bmod N}$     // message transmission to $P_{i+1}$
10:         $P_i$ computes $\mathbf{a}_i^j \leftarrow \mathbf{a}_{(i-1) \bmod N}^{j-1}\mathbf{P}_i$   // private computation of vector
11:     end for
12: end for
Ensure: The value of $\mathbf{a}_i^{N-1}$ is the mutual key which is identical for every participant $P_i$ $(0 \leq i \leq N-1)$.
```

The generic key distribution procedure in Table III uses $N-1$ rounds of messages transmission to establish a mutual vector for N group members. This protocol also requires a prearranged initial circular configuration before the key exchange occurs. The configuration is compulsory because every member in the group always transmits its messages to the same predetermined recipient at any round. From Table III, this configuration also implies that each member performs uniform steps in each of the rounds. By observation, each member sends $N-1$ messages, performs N right-multiplications, and receives $N-1$ messages in the entire process. Additionally, the scheme also infers that the total number of right-multiplications in the entire key exchange is N^2. We summarize some of the important characteristics in this protocol as follows:

1) total rounds: $N-1$
2) total messages sent per member: $N-1$
3) total messages received per member: $N-1$
4) total right-multiplications (vector-matrix multiplications) per member: N
5) total messages in the entire protocol: $N(N-1)$
6) total right-multiplications in the entire protocol: N^2.

B. Megrelishvili Key Distribution with Upflow-Downflow Rounds

The protocol described in Table III has several drawbacks. Firstly, it needs an *a priori* ordering of the group members. Among N members $P_0, P_1, \ldots, P_{N-1}$, this ordering is required because participant P_i always sends its messages to participant $P_{(i+1) \bmod N}$. Secondly, this protocol needs $N-1$ rounds of key exchange to establish the mutual secret vector. This section discusses a Megrelishvili key distribution which uses only two rounds, i.e., the upflow and downflow rounds, which was first proposed in [1]. This procedure is adapted from the GDH.1 scheme for the Diffie-Hellman key distribution explained in [5]. The public parameters used in this protocol is identical to those used in the previous one.

The aim of the upflow stage is to gather the contribution of each participant. Initially, for each $0 \leq i \leq N-1$, participant P_i constructs the private matrix $\mathbf{P}_i$. The upflow round is initiated by P_0 by computing the vectors $\mathbf{v}\mathbf{P}_0$ and sending the *upflow list* $\mathsf{U} = [\mathbf{v}\mathbf{P}_0]$ to P_1. Participant P_1 then right-multiplies the last value in U by $\mathbf{P}_1$ (i.e., calculating $\mathbf{v}\mathbf{P}_0\mathbf{P}_1$) and updates the upflow list by appending the resulting vector (i.e., $\mathbf{v}\mathbf{P}_0\mathbf{P}_1$) to U. Next, P_1 sends the updated upflow list $\mathsf{U} = [\mathbf{v}\mathbf{P}_0, \mathbf{v}\mathbf{P}_0\mathbf{P}_1]$ to P_2. In general, for each $1 \leq i \leq N-2$, participant P_i right-multiplies the last value in U (i.e., $\mathbf{v}\prod_{k=0}^{i-1}\mathbf{P}_k$) by $\mathbf{P}_i$ and attaches the result to the previous list. This implies that each participant P_i with $0 \leq i \leq N-2$ always sends the upflow list $\mathsf{U} = \left[\mathbf{v}\prod_{k=0}^{j}\mathbf{P}_k : 0 \leq j \leq i\right]$ to participant P_{i+1}. At the end of the upflow stage, participant P_{N-1} receives the upflow list $\mathsf{U} = \left[\mathbf{v}\prod_{k=0}^{j}\mathbf{P}_k : 0 \leq j \leq N-2\right]$. The mutual key $\mathbf{v}\prod_{k=0}^{N-1}\mathbf{P}_k$ can be retrieved by P_{N-1} simply by right-multiplying the last element of U by $\mathbf{P}_{N-1}$.

The purpose of the downflow stage is to distribute *appropriate vector* for each participant so that each of them can compute the mutual secret key. The downflow stage consists of successive message transmissions from P_i to P_{i-1} for $i \in [1, N-1]$ in reverse order. Participant P_{N-1} initiates the downflow stage by constructing the initial *downflow list* D, where

$$\mathsf{D} = \left[\mathbf{v}\left(\prod_{k=N-1}^{N-1}\mathbf{P}_k\right)\left(\prod_{k=0}^{j-1}\mathbf{P}_k\right) : 0 \leq j \leq N-2\right] \tag{6}$$

$$= \left[\mathbf{v}\mathbf{P}_{N-1}\left(\prod_{k=0}^{j-1}\mathbf{P}_k\right) : 0 \leq j \leq N-2\right] \tag{7}$$

$$= \left[\mathbf{v}\mathbf{P}_{N-1}, \mathbf{v}\mathbf{P}_{N-1}\mathbf{P}_0, \mathbf{v}\mathbf{P}_{N-1}\mathbf{P}_0\mathbf{P}_1, \ldots, \mathbf{v}\mathbf{P}_{N-1}\mathbf{P}_0\mathbf{P}_1 \cdots \mathbf{P}_{N-4}\mathbf{P}_{N-3}\right], \tag{8}$$

and then sends this list to P_{N-2}. After that, P_{N-2} pops the last element of D and right-multiplies this element by $\mathbf{P}_{N-2}$ to retrieve the key. Observe that

$$\mathbf{v}\mathbf{P}_{N-1}\left(\prod_{k=0}^{N-3}\mathbf{P}_k\right)\mathbf{P}_{N-2} = \mathbf{v}\left(\prod_{k=0}^{N-3}\mathbf{P}_k\right)\mathbf{P}_{N-2}\mathbf{P}_{N-1} = \mathbf{v}\prod_{k=0}^{N-1}\mathbf{P}_k \tag{9}$$

due to the commutativity of the private matrices' product. Next, P_{N-2} updates the list by right-multiplying each of the remaining $N-2$ vectors by $\mathbf{P}_{N-2}$, yielding a new downflow list

$$\mathsf{D} = \left[\mathbf{v}\left(\prod_{k=N-2}^{N-1}\mathbf{P}_k\right)\left(\prod_{k=0}^{j-1}\mathbf{P}_k\right) : 0 \leq j \leq N-3\right] \tag{10}$$

$$= \left[\mathbf{v}\mathbf{P}_{N-2}\mathbf{P}_{N-1}\left(\prod_{k=0}^{j-1}\mathbf{P}_k\right) : 0 \leq j \leq N-3\right] \tag{11}$$

$$= \left[\mathbf{v}\mathbf{P}_{N-2}\mathbf{P}_{N-1}, \mathbf{v}\mathbf{P}_{N-2}\mathbf{P}_{N-1}\mathbf{P}_0, \ldots, \mathbf{v}\mathbf{P}_{N-2}\mathbf{P}_{N-1}\mathbf{P}_0 \cdots \mathbf{P}_{N-5}\mathbf{P}_{N-4}\right]. \tag{12}$$

In general, for $i \in [1, N-2]$ in reverse order, P_i pops the last element of downflow list D from P_{i+1}, i.e.,$\mathbf{v}\left(\prod_{k=i+1}^{N-1}\mathbf{P}_k\right)\left(\prod_{k=0}^{i-1}\mathbf{P}_k\right)$ and retrieves the mutual key by right-multiplying this vector by $\mathbf{P}_i$, notice that

$$\mathbf{v}\left(\prod_{k=i+1}^{N-1}\mathbf{P}_k\right)\left(\prod_{k=0}^{i-1}\mathbf{P}_k\right)\mathbf{P}_i = \mathbf{v}\left(\prod_{k=0}^{i-1}\mathbf{P}_k\right)\mathbf{P}_i\left(\prod_{k=i+1}^{N-1}\mathbf{P}_k\right) \tag{13}$$

$$= \mathbf{v}\prod_{k=0}^{N-1}\mathbf{P}_k, \tag{14}$$

the equality in (13) follows from the commutativity of the matrices' product. Subsequently, P_i updates D by right-multiplying the remaining i vectors by $\mathbf{P}_i$, yielding the list

$$\mathsf{D} = \left[\mathbf{v}\left(\prod_{k=i}^{N-1}\mathbf{P}_k\right)\left(\prod_{k=0}^{j-1}\mathbf{P}_k\right) : 0 \leq j \leq i-1\right], \tag{15}$$

and sends this list to P_{i-1}. At the end of downflow round, P_0 obtains the following list D from P_1

$$\mathsf{D} = \left[\mathbf{v}\left(\prod_{k=1}^{N-1}\mathbf{P}_k\right)\left(\prod_{k=0}^{j-1}\mathbf{P}_k\right) : 0 \leq j \leq 0\right] = \left[\mathbf{v}\prod_{k=1}^{N-1}\mathbf{P}_k\right]. \tag{16}$$

To get the mutual secret vector, P_0 simply right-multiplies the only element in D by $\mathbf{P}_0$. We summarize this protocol in Table IV and present its simulation procedure in Algorithm 2.

Setup for public parameters
The public parameters are identical to those described for the two-party protocol.
Generation of the private matrices
The procedure for generating the private matrices is identical to the procedure described in Table III.
Upflow round
Initialization for the upflow list P_0 constructs the list $\mathsf{U} = [\mathbf{v}\mathbf{P}_0]$ and sends U to P_1
For all $1 \leq i \leq N-2$, (i) P_i reads the last element of U and right-multiplies it by $\mathbf{P}_i$ (ii) P_i appends the right-multiplication result in (i) to U (iii) P_i transmits U to P_{i+1}
Mutual key retrieval for P_{N-1}
P_{N-1} reads the last element of U and right-multiplies it by $\mathbf{P}_{N-1}$, the result is the mutual secret key (see (13) and (14)).
Downflow rounds
Initialization for the downflow list P_{N-1} constructs the initial downflow list as in (7) and sends D to P_{N-2}
For all $0 \leq i \leq N-2$ in reverse order (i) P_i pops the last element of D and right-multiplies it by $\mathbf{P}_i$, the result is the mutual secret key (see see (13) and (14)) (ii) if $i \neq 0$, P_i right-multiplies each of the remaining i elements in D by $\mathbf{P}_i$ and transmits D to P_{i-1}

TABLE IV: Megrelishvili group key distribution with upflow-downflow rounds as in [1].

Algorithm 2 A procedure for simulating the Megrelishvili key distribution using upflow-downflow rounds as in [1].

```
Require: Public parameters as explained in Table IV and an integer $N \geq 2$ which denotes the number of group members.
 1: for $i \leftarrow 0$ to $N-1$ do                                  // $N$ group members
 2:     $\mathbf{P}_i \leftarrow \mathbf{M}^{\alpha_i}$              // private matrices generation for $N$ group members
 3: end for
 4: $\mathsf{U} \leftarrow [\mathbf{v}\mathbf{P}_0]$                  // initialization of the upflow list
 5: $P_0$ transmits $\mathsf{U}$ to $P_1$
 6: for $i \leftarrow 1$ to $N-2$ do                                  // upflow round
 7:     $\mathsf{U} \leftarrow \mathsf{U}.\mathsf{append}\left(\mathsf{U}[i-1] \cdot \mathbf{A}_i\right)$   // upflow list construction
 8:     $P_i$ sends $\mathsf{U}$ to $P_{i+1}$
 9: end for
10: key for $P_{N-1}$ is $\mathsf{U}[N-2] \cdot \mathbf{P}_{N-1}$      // key retrieval for $P_{N-1}$
11: $P_{N-1}$ constructs $\mathsf{D}$ as in Equation (7) and transmits $\mathsf{D}$ to $P_{N-1}$
12: for $i \leftarrow N-2$ down to $0$ do                             // downflow round
13:     $P_i$ pops $\mathsf{D}[\mathsf{D}.\mathsf{length}-1]$          // popping the last element of the list
14:     key for $P_i$ is $\mathsf{D}[\mathsf{D}.\mathsf{length}-1] \cdot \mathbf{P}_i$   // key retrieval for $P_i$
15:     if $i \neq 0$ then
16:         $P_i$ right-multiplies each of the remaining elements in $\mathsf{D}$ by $\mathbf{P}_i$ and sends $\mathsf{D}$ to $P_{i-1}$
17:     end if
18: end for
Ensure: Each participant has an identical secret key.
```

Unlike the generic multi-party Megrelishvili key distribution scheme in Section IV-A, the Megrelishvili key distribution procedure with upflow-downflow rounds does not require a predefined circular ordering of the group members. Moreover, each participant in this upflow-downflow scheme performs different computation depending on its order in a linear configuration of the members. This property also allows the completion of the key distribution in two rounds.

The correctness of this protocol is derived from the property that the matrices $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{N-1}$ commute. This implies that the right-multiplication procedure in (13) always produces the same vector for every member of the group. The reader is referred to [1, Theorem 2] for the formal proof regarding the correctness of Megrelishvili protocol with upflow-downflow rounds.

The upflow-downflow scheme allows a distribution of the mutual secret key using only two rounds, i.e., the upflow and downflow rounds, regardless the number of participants within the group. Moreover, unlike the generic scheme in Section IV-A, this scheme does not need a prior circular synchronization of the group members. However, each group member performs different computational procedure that depends on its appearance during the upflow round. During the upflow stage, P_i performs one vector-matrix multiplication and updates the previous list with the resulting value. At the end of the upflow round, the last participant retrieves the mutual key by right-multiplying the last element in the upflow list by its own private matrix. Consequently, there are N vector-matrix multiplications in total during the upflow round and the key retrieval for P_{N-1}.

The downflow stage consists of sequential messages transmission in reverse order from that of the upflow stage. From Table IV, we know that each P_i for $1 \leq i \leq N-1$ performs i vector-matrix multiplications before it sends the downflow list to P_{i-1}. In addition, each P_i for $0 \leq i \leq N-2$ performs a vector-matrix multiplication to retrieve the mutual key. Thus, each P_i for $0 \leq i \leq N-2$ performs $i+1$ right-multiplications during the downflow round, and consequently the total number of right-multiplications performed by all P_i for $0 \leq i \leq N-2$ in this round is $\sum_{i=0}^{N-2}(i+1) = N(N-1)/2$. Since P_{N-1} performs $N-1$ right-multiplications at the beginning of the downflow stage, the total number of right-multiplications performed by all participants during the downflow stage is $N(N-1)/2 + (N-1) = \left(N^2+N-2\right)/2$.

From the previous analysis, the total number of right-multiplications within the protocol described in

Table IV is

$$N+\frac{1}{2}N^2+\frac{1}{2}N-1=\frac{1}{2}N^2+\frac{3}{2}N-1=\frac{1}{2}\left(N^2+3N-2\right), \tag{17}$$

which is $\left(N^2-3N+2\right)/2$ multiplications fewer than that in the generic procedure in Section IV-A. In summary, the Megrelishvili key distribution with upflow-downflow scheme has the following computational properties:

1) total rounds: 2 (upflow and downflow)
2) total messages sent per member: 2 for P_i with $0<i<N-1$ (each contains $i+1$ vectors during the upflow stage and i vectors during the downflow stage); 1 for P_0 and P_{N-1} (P_0 sends one vector while P_{N-1} sends $N-1$ vectors)
3) total messages received per member: 2 for P_i with $0<i<N-1$ (each contains i vectors during the upflow stage and $i+1$ vectors during the downflow stage); 1 for P_0 and P_{N-1} (P_0 receives one vector while P_{N-1} receives $N-1$ vectors)
4) total right-multiplications (vector-matrix multiplications) per member: $i+2$ for P_i with $0\leq i<N-1$; N for P_{N-1}
5) total messages in the entire protocol: $2\left(N-1\right)$ with varying size from one to $N-1$ vectors
6) total right-multiplications in the entire protocol: $(N^2+3N-2)/2$.

C. Megrelishvili Key Distribution with Upflow-Broadcast Rounds

The key distribution protocol with upflow-downflow rounds in Section IV-B allows a group to agree on a mutual secret key using only two rounds of messages transmission. The first round is the upflow round whose purpose is to collect the contribution of each group member. The second round is the downflow round whose objective is to distribute vectors to every participant. Each round contains a sequence of messages transmission between two adjacent participants. Suppose we consider a group of N members $P_0,P_1,\ldots,P_{N-1}$. During the upflow round P_i always transmits its messages to P_{i+1} for $0\leq i\leq N-2$. The downflow round works in reverse direction to that of the upflow round, i.e., participant P_i always sends its messages to P_{i-1} for $1\leq i\leq N-1$ in reverse order. Unlike the generic Megrelishvili key distribution scheme in Section IV-A, the participants in the key distribution with upflow-downflow scheme cannot retrieve the mutual secret key simultaneously. From observation, the key retrieval must be done sequentially from P_{N-1} to P_0 with P_i is the $(N-i)$-th participant to retrieve the key. That is, the first member to retrieve the key is P_{N-1}, whereas the last member to retrieve the key is P_0.

In this section, we propose a two-stage Megrelishvili key distribution that allows a simultaneous key retrieval for almost all of the group members. This scheme is adapted from GDH.2 scheme for the Diffie-Hellman key distribution with upflow and broadcast rounds described in [5]. The first stage is the upflow stage that is similar to the upflow round explained in the previous protocol. The aim of this stage is to collect the contribution of each participant. At the end of this stage P_{N-1} retrieves the mutual key by right-multiplying a particular vector by its private matrix. The second stage is the broadcast stage whose aim is to distribute appropriate vector to each participant. Unlike the downflow round in Section IV-B, the distribution of the vectors is performed by broadcast method. In this stage, P_{N-1} constructs a list B of $N-1$ vectors and then broadcasts this list to all other group members. Every member except P_{N-1} uses the list B to simultaneously perform a specific vector-matrix multiplication for retrieving the mutual secret vector.

Suppose there are N participants $P_0,P_1,\ldots,P_{N-1}$ whose corresponding private matrices are $\mathbf{P}_0$, $\mathbf{P}_1$, $\ldots$, $\mathbf{P}_{N-1}$, respectively. To commence the upflow stage, P_0 creates the upflow list $\mathsf{U}=[\mathbf{vP}_0]$ and sends this list to P_1. Next, P_1 uses the list U to construct the updated list $\mathsf{U}=[\mathbf{vP}_0\mathbf{P}_1,\mathbf{vP}_0,\mathbf{vP}_1]$ and subsequently sends U to P_2. Participant P_2 then constructs the updated list U of four elements where the first element of the new U is the first element in the previous list right-multiplied by $\mathbf{P}_2$, the second element of the new U is the first element in the previous list, and the two remaining elements of the new U are respectively the third and the last elements in the previous U right-multiplied by $\mathbf{P}_2$. In other words, P_2 produces the updated list $\mathsf{U}=[\mathbf{vP}_0\mathbf{P}_1\mathbf{P}_2,\mathbf{vP}_0\mathbf{P}_1,\mathbf{vP}_0\mathbf{P}_2,\mathbf{vP}_1\mathbf{P}_2]$ and subsequently sends U to P_3. In general, suppose P_i where $2\leq i\leq N-2$ receives the upflow list U_{old} of length $i+1$ from

P_{i-1}. Participant P_i uses U_{old} to create the updated list U_{new} of length $i+2$ using the following rules:

$$\mathsf{U}_{\text{new}}[0] = \mathsf{U}_{\text{old}}[0] \cdot \mathbf{P}_i, \tag{18}$$

$$\mathsf{U}_{\text{new}}[1] = \mathsf{U}_{\text{old}}[0], \tag{19}$$

$$\mathsf{U}_{\text{new}}[j] = \mathsf{U}_{\text{old}}[j-1] \cdot \mathbf{P}_i \text{ for all } 2 \le j \le i+1. \tag{20}$$

In other words, if P_{i-1} sends the upflow list $\mathsf{U}_{\text{old}} = [\mathsf{U}_{\text{old}}[0], \mathsf{U}_{\text{old}}[1], \ldots, \mathsf{U}_{\text{old}}[i]]$ to P_i, then P_i creates the updated upflow list $\mathsf{U}_{\text{new}} = [\mathsf{U}_{\text{old}}[0] \cdot \mathbf{P}_i, \mathsf{U}_{\text{old}}[0], \mathsf{U}_{\text{old}}[1] \cdot \mathbf{P}_i, \ldots, \mathsf{U}_{\text{old}}[i] \cdot \mathbf{P}_i]$. We now prove the following lemma.

Lemma 1 *For any $2 \le i \le N-1$, participant P_i receives the list U of length $i+1$ from P_{i-1} where*

$$\mathsf{U} = \left[\mathbf{v}\left(\prod_{k=0}^{i-1}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq i-1}^{i-1}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq i-2}^{i-1}\mathbf{P}_k\right), \ldots, \mathbf{v}\left(\prod_{k=0,k\neq 0}^{i-1}\mathbf{P}_k\right)\right] \tag{21}$$

Proof: We prove the lemma by induction on i and we use the rules (18), (19), and (20). For $i=2$, (21) tells us that P_2 receives the following list U from P_1

$$\mathsf{U} = \left[\mathbf{v}\left(\prod_{k=0}^{1}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq 1}^{1}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq 0}^{1}\mathbf{P}_k\right)\right] = [\mathbf{v}\mathbf{P}_0\mathbf{P}_1, \mathbf{v}\mathbf{P}_0, \mathbf{v}\mathbf{P}_1], \tag{22}$$

which conforms to the aforementioned description that P_1 sends $\mathsf{U} = [\mathbf{v}\mathbf{P}_0\mathbf{P}_1, \mathbf{v}\mathbf{P}_0, \mathbf{v}\mathbf{P}_1]$ to P_2. Assume that (21) holds for P_{i-1}, that is, P_{i-1} receives the following list U_{old} from P_{i-2}

$$\mathsf{U}_{\text{old}} = \left[\mathbf{v}\left(\prod_{k=0}^{i-2}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq i-2}^{i-2}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq i-3}^{i-2}\mathbf{P}_k\right), \ldots, \left(\mathbf{v}\prod_{k=0,k\neq 0}^{i-2}\mathbf{P}_k\right)\right]. \tag{23}$$

Notice that for $1 \le j \le i-1$ we have $\mathsf{U}_{\text{old}}[j] = \mathbf{v}\left(\prod_{k=0,k\neq i-j-1}^{i-2}\mathbf{P}_k\right)$. We shall prove that (21) holds for P_i. Suppose U_{new} is the updated list sent by P_{i-1} to P_i. By rules (18), (19), and (20), we have

1) $\mathsf{U}_{\text{new}}[0] = \mathsf{U}_{\text{old}}[0] \cdot \mathbf{P}_{i-1} = \mathbf{v}\left(\prod_{k=0}^{i-2}\mathbf{P}_k\right)\mathbf{P}_{i-1} = \mathbf{v}\left(\prod_{k=0}^{i-1}\mathbf{P}_k\right)$,
2) $\mathsf{U}_{\text{new}}[1] = \mathsf{U}_{\text{old}}[0] = \mathbf{v}\left(\prod_{k=0}^{i-2}\mathbf{P}_k\right) = \mathbf{v}\left(\prod_{k=0,k\neq i-1}^{i-1}\mathbf{P}_k\right)$,
3) for $2 \le j \le i$, $\mathsf{U}_{\text{new}}[j] = \mathsf{U}_{\text{old}}[j-1] \cdot \mathbf{P}_{i-1} = \mathbf{v}\left(\prod_{k=0,k\neq i-j}^{i-2}\mathbf{P}_k\right)\mathbf{P}_{i-1} = \mathbf{v}\left(\prod_{k=0,k\neq i-j}^{i-1}\mathbf{P}_k\right)$.

Consequently, we obtain

$$\mathsf{U}_{\text{new}} = \left[\mathbf{v}\left(\prod_{k=0}^{i-1}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq i-1}^{i-1}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq i-2}^{i-1}\mathbf{P}_k\right), \ldots, \mathbf{v}\left(\prod_{k=0,k\neq 0}^{i-1}\mathbf{P}_k\right)\right] \tag{24}$$

which is consistent to (21), and thus the proof is complete. □

From Lemma 1, we infer that P_{N-1} receives the list

$$\mathsf{U} = \left[\mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq N-2}^{N-2}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq N-3}^{N-2}\mathbf{P}_k\right), \ldots, \mathbf{v}\left(\prod_{k=0,k\neq 0}^{N-2}\mathbf{P}_k\right)\right] \tag{25}$$

from P_{N-2}. To get the mutual key, P_{N-1} simply right-multiplies $\mathsf{U}[0]$ by $\mathbf{P}_{N-1}$. Afterward, P_{N-1} initiates the broadcast round by constructing the broadcast list B of $N-1$ vectors. To construct this list, P_{N-1} initially removes $\mathsf{U}[0]$ and shifts all remaining entries one position to the left. Next, P_{N-1} right-multiplies each of the entries by its private matrix. More formally, the entry $\mathsf{B}[j]$ of the broadcast list $\mathsf{B} = [\mathsf{B}[0], \mathsf{B}[1] \ldots, \mathsf{B}[N-2]]$ of length $N-1$ is defined as

$$\mathsf{B}[j] = \mathsf{U}[j+1] \cdot \mathbf{P}_{N-1} \text{ for all } 0 \le j \le N-2. \tag{26}$$

The broadcast list B is transmitted by P_{N-1} to all remaining members simultaneously. The mutual key retrieval by P_i for $0 \le i \le N-2$ is performed by calculating the vector $\mathsf{B}[N-2-i] \cdot \mathbf{P}_i$. In practice, P_{N-1} may send unique vector that correlates to the mutual key computation for each of the other members. That is, P_{N-1} sends only the value $\mathsf{B}\left[N-2-i\right]$ to each participant P_i for $0 \leq i \leq N-2$. We present the summary of this protocol in Table V and its simulation procedure in Algorithm 3. The correctness of this key distribution scheme is derived from the property that the product $\prod_{k\in[0,N-1]} \mathbf{P}_k$ is always the same regardless the order of the matrices. This condition happens because the private matrices $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{N-1}$ commute. We prove the correctness of this protocol in Theorem 1.

Theorem 1 *At the end of the broadcast round, each group member in the upflow-broadcast Megrelishvili key distribution scheme described in Table V gets an identical vector as its mutual secret key.*

Proof: By Lemma 1, P_{N-1} receives the upflow list U from P_{N-2} with $\mathsf{U}\left[0\right] = \mathbf{v}\left(\prod_{k=0}^{N-2} \mathbf{P}_k\right)$. Subsequently, P_{N-1} right-multiplies $\mathsf{U}\left[0\right]$ by $\mathbf{P}_{N-1}$ to obtain the mutual key, i.e.,

$$\mathsf{U}\left[0\right] \cdot \mathbf{P}_{N-1} = \mathbf{v}\left(\prod_{k=0}^{N-2} \mathbf{P}_k\right)\mathbf{P}_{N-1} = \mathbf{v}\prod_{k=0}^{N-1} \mathbf{P}_k. \tag{27}$$

Next P_{N-1} constructs the broadcast list B using (26). By observation, we have

$$\mathsf{B} = \left[\mathbf{v}\left(\prod_{k=0,k\neq N-2-j}^{N-1} \mathbf{P}_k\right) : 0 \leq j \leq N-2\right] \tag{28}$$

$$= \left[\mathbf{v}\left(\prod_{k=0,k\neq N-2}^{N-1} \mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq N-3}^{N-1} \mathbf{P}_k\right), \ldots, \mathbf{v}\left(\prod_{k=0,k\neq 1}^{N-1} \mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq 0}^{N-1} \mathbf{P}_k\right)\right]. \tag{29}$$

Each participant P_i $(0 \leq i \leq N-2)$ retrieves the mutual secret vector by computing $\mathsf{B}\left[N-2-i\right] \cdot \mathbf{P}_i$, i.e.,

$$\mathsf{B}\left[N-2-i\right] \cdot \mathbf{P}_i = \mathbf{v}\left(\prod_{k=0,k\neq N-2-(N-2-i)}^{N-1} \mathbf{P}_k\right) \cdot \mathbf{P}_i = \mathbf{v}\left(\prod_{k=0,k\neq i}^{N-1} \mathbf{P}_k\right) \cdot \mathbf{P}_i \tag{30}$$

$$= \mathbf{v}\left(\prod_{k=0}^{N-1} \mathbf{P}_k\right), \tag{31}$$

the equality in (31) comes from the commutativity of matrices' product. Therefore, we conclude that each member P_i $(0 \leq i \leq N-1)$ gets identical mutual vector at the end of the broadcast round. □

The upflow-broadcast scheme combines the benefits of the generic scheme in Section IV-A and the upflow-downflow scheme in Section IV-B. Specifically, it provides a two stage key distribution whilst allows a simultaneous mutual key retrieval for almost all participants. Additionally, the upflow-broadcast scheme does not need a predetermined configuration of the group members before the key distribution occurs. However, the computational burden for each participant is different and it depends on the participant's order during the upflow stage.

By observation, each participant P_i $(0 \leq i \leq N-2)$ carries out $i+1$ vector-matrix multiplications during the upflow round. Thus, the total number of this operation in the upflow round is $\sum_{i=0}^{N-2}(i+1) = N\left(N-1\right)/2$. For the next step, participant P_{N-1} performs one right-multiplication for retrieving the mutual key and $N-1$ right-multiplications for constructing the broadcast list. Finally, each participant P_i $(0 \leq i \leq N-2)$ retrieves the mutual key by performing one vector-matrix multiplication. Thus, the total number of right-multiplications for the mutual key retrieval in the entire procedure is N. Therefore, the total number of vector-matrix multiplications in the entire scheme is

$$N\left(N-1\right)/2 + \left(N-1\right) + N = \frac{1}{2}N^2 + \frac{3}{2}N - 1 = \left(N^2 - 3N + 2\right)/2, \tag{32}$$

which is equal to (17). This implies that the total number of operations in upflow-broadcast scheme is identical to that of the upflow-downflow scheme. The Megrelishvili key distribution with upflow-broadcast scheme has the following characteristics:

1) total rounds: 2 (upflow and broadcast)

Setup for public parameters
The public parameters are identical to those described for the two-party protocol.
Generation of the private matrices
The procedure for generating the private matrices is identical to the procedure described in Table III.
Upflow round
Initialization for the upflow list
P_0 constructs the list $\mathsf{U} = [\mathbf{v}\mathbf{P}_0]$ and sends U to P_1 P_1 constructs the list $\mathsf{U} = [\mathbf{v}\mathbf{P}_0\mathbf{P}_1, \mathbf{v}\mathbf{P}_0, \mathbf{v}\mathbf{P}_1]$ and sends U to P_2
For all $2 \leq i \leq N-2$,
(i) P_i receives the list U_{old} of length $i+1$ from P_{i-1} (ii) P_i constructs the updated list U_{new} of length $i+2$ using the rules (18), (19), and (20), that is: $\mathsf{U}_{\text{new}}[1] = \mathsf{U}_{\text{old}}[0]$, $\mathsf{U}_{\text{new}}[0] = \mathsf{U}_{\text{old}}[0] \cdot \mathbf{P}_i$, and $\mathsf{U}_{\text{new}}[j] = \mathsf{U}_{\text{old}}[j-1] \cdot \mathbf{P}_i$ for all $2 \leq j \leq i+1$
Mutual key retrieval for P_{N-1}
P_{N-1} reads the first element of U and right-multiplies it by $\mathbf{P}_{N-1}$, the operation $\mathsf{U}[0] \cdot \mathbf{P}_{N-1}$ yields the mutual secret vector for P_{N-1}
Broadcast round
Initialization for the broadcast list
P_{N-1} constructs the initial broadcast list as in (26), that is, the broadcast list is B where $\mathsf{B}[j] = \mathsf{U}[j+1] \cdot \mathbf{P}_{N-1}$ for all $0 \leq j \leq N-2$.
Mutual key retrieval for P_i $(0 \leq i \leq N-2)$
P_i computes $\mathsf{B}[N-2-i] \cdot \mathbf{P}_i$ to retrieve the mutual secret key.

TABLE V: Megrelishvili group key distribution with upflow-broadcast rounds.

2) total messages transmitted per member: 1 (each contains different number of vectors: P_0 sends one vector, P_N sends $N-1$ vectors, and P_i for $1 \leq i \leq N-2$ sends $i+2$ vectors)
3) total messages received per member: 2 for P_i with $0 < i < N-1$ (one during the upflow round and one during the broadcast round, each P_i where $2 \leq i \leq N-2$ receives $i+1$ vectors during the upflow round, P_1 receives one vector during the upflow round, and each P_i where $1 \leq i \leq N-2$ receives $N-1$ vectors during the broadcast round); 1 for P_0 and P_{N-1} (P_0 receives $N-1$ vectors while P_{N-1} receives N vectors)
4) total right-multiplications (vector-matrix multiplications) per member: $i+2$ for P_i with $0 \leq i < N-1$; N for P_{N-1}
5) total messages in the entire protocol: N with varying size from one to N vectors
6) total right-multiplications in the entire protocol: $(N^2+3N-2)/2$.

D. Megrelishvili Key Distribution with Upflow-Broadcast-Response Rounds

The vector-matrix multiplication (i.e., right-multiplication of a vector $\mathbf{v}$ by a matrix $\mathbf{P}$) is the most extensively used mathematical operation in Megrelishvili key distribution schemes. The effectiveness of each scheme can be measured by the number of vector-matrix multiplications in the entire protocol. The generic key distribution scheme in Section IV-A requires N^2 vector-matrix multiplications, whereas both key distribution procedures in Section IV-B and Section IV-C need $(N^2+3N-2)/2$ vector-matrix multiplications. We deduce that each of these schemes requires $\mathcal{O}\left(N^2\right)$ number of vector-matrix multiplications in its whole process. In other words, the number of vector-matrix multiplications required for completing the key distribution procedure is quadratic with respect to the group size.

In this section, we propose a version of Megrelishvili key distribution for N participants that requires no more than $\mathcal{O}(N)$ number of vector-matrix multiplications. In other words, the number of vector-matrix multiplications in our proposed scheme is linear as regards to the size of the group. This scheme is adapted from GDH.3 protocol for the group Diffie-Hellman key exchange explained in [5]. The scheme consists of four rounds: the upflow round, the first broadcast round, the response round, and the second broadcast round. Henceforth, we refer to this scheme as Megrelishvili key distribution with upflow-broadcast-response rounds. Similar to the previous key distribution procedures, the correctness of this scheme relies on the commutativity of the private matrices' product.

We consider a group of N participants labeled as P_i for $0 \leq i \leq N-1$, each with corresponding private matrix $\mathbf{P}_i$. The upflow round in this scheme is simpler than that of the upflow-downflow scheme

Algorithm 3 A procedure for simulating the Megrelishvili key distribution using upflow-broadcast rounds.

```
Require: Public parameters as explained in Table V and an integer $N \geq 3$ which denotes the number of group members.
 1: for $i \leftarrow 0$ to $N-1$ do                                  // $N$ group members
 2:     $\mathbf{P}_i \leftarrow \mathbf{M}^{\alpha_i}$              // private matrices generation for $N$ group members
 3: end for
 4: $\mathsf{U} \leftarrow [\mathbf{vP}_0]$                          // initialization of the upflow list
 5: $P_0$ transmits $\mathsf{U}$ to $P_1$
 6: $\mathsf{U} \leftarrow [\mathbf{vP}_0\mathbf{P}_1, \mathbf{vP}_0, \mathbf{vP}_1]$
 7: $P_1$ transmits $\mathsf{U}$ to $P_2$
 8: for $i \leftarrow 2$ to $N-2$ do                                  // upflow round
 9:     $\mathsf{U}_{\text{old}} \leftarrow \mathsf{U}$                // storing the previous upflow list
10:     $\mathsf{U}[0] \leftarrow \mathsf{U}_{\text{old}}[1] \cdot \mathbf{P}_i$      // applying (18)
11:     $\mathsf{U}[1] \leftarrow \mathsf{U}_{\text{old}}[0]$                       // applying (19)
12:     for $j \leftarrow 2$ to $i$ do                                // applying (20)
13:         $\mathsf{U}[j] \leftarrow \mathsf{U}_{\text{old}}[j-1] \cdot \mathbf{P}_i$
14:     end for
15:     $\mathsf{U} \leftarrow \mathsf{U}.\mathsf{append}(\mathsf{U}_{\text{old}}[i] \cdot \mathbf{P}_i)$   // handling the case $j = i+1$ in (20)
16:     $P_i$ sends $\mathsf{U}$ to $P_{i+1}$
17: end for
18: key for $P_{N-1}$ is $\mathsf{U}[0] \cdot \mathbf{P}_{N-1}$                // key retrieval for $P_{N-1}$
19: $P_{N-1}$ constructs $\mathsf{B}$ as in (26) and broadcasts $\mathsf{B}$ to all other group members   // broadcast round
20: for all $i \in [0, N-2]$ do
21:     key for $P_i$ is $\mathsf{B}[N-2-i] \cdot \mathbf{P}_i$        // key retrieval for $P_i$ $(0 \leq i \leq N-2)$
22: end for
Ensure: Each participant has an identical secret key.
```

in Section IV-B. Initially, P_0 sends $\mathbf{vP}_0$ to P_1 who then uses this value to send $\mathbf{vP}_0\mathbf{P}_1$ to P_2. In general, for $1 \leq i \leq N-2$, P_i right-multiplies the vector from P_{i-1} by $\mathbf{P}_i$ and sends the result to P_{i+1}. Thus, each participant P_i $(0 \leq i \leq N-2)$ sends the vector $\mathbf{v}\prod_{k=0}^{i}\mathbf{P}_k$ to P_{i+1}.

Apart from sending the vector $\mathbf{v}\prod_{k=0}^{N-2}\mathbf{P}_k$ to P_{N-1}, participant P_{N-2} also broadcasts this vector to all other participant P_i $(0 \leq i \leq N-3)$. Therefore, every P_i $(0 \leq i \leq N-1)$ possesses the value $\mathbf{v}\prod_{k=0}^{N-2}\mathbf{P}_k$. Notice that as early as the upflow round is completed, P_{N-1} can retrieve the mutual key by right-multiplying the vector from P_{N-2} by its own private matrix.

The next stage is the response round where each participant P_i $(0 \leq i \leq N-2)$ calculates the following vector

$$\mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\mathbf{P}_i^{-1} = \mathbf{v}\prod_{k=0,k\neq i}^{N-2}\mathbf{P}_k. \tag{33}$$

Recall that each $\mathbf{P}_i$ is invertible (thus, its inverse exists) and the private matrices commute. In addition, the computation of $\mathbf{P}_i^{-1}$ can be performed using $\mathcal{O}\left(n^3\right)$ scalar operations in $\mathbb{F}_q$ whenever $\mathbf{P}_i$ is an $n \times n$ nonsingular matrix over $\mathbb{F}_q$. Afterward, each P_i $(0 \leq i \leq N-2)$ defines $\mathsf{R}[i] = \mathbf{v}\prod_{k=0,k\neq i}^{N-2}\mathbf{P}_k$ and transmits this value to P_{N-1}. The last step implies P_{N-1} receives the values $\mathsf{R}[i]$ for all $0 \leq i \leq N-2$. These values are then stored in the *response list* $\mathsf{R} = [\mathsf{R}[0], \mathsf{R}[1], \ldots, \mathsf{R}[N-2]]$ of length $N-1$.

After constructing the response list R, participant P_{N-1} creates the broadcast list B of length $N-1$ where $\mathsf{B}[j] = \mathsf{R}[j] \cdot \mathbf{P}_{N-1}$ for each $0 \leq j \leq N-2$. Subsequently, P_{N-1} broadcasts B to all other remaining members simultaneously. However, in practice P_{N-1} can send unique value to each of the remaining participants, i.e., P_{N-1} sends each P_i $(0 \leq i \leq N-2)$ the vector $\mathsf{B}[i]$.

To conclude the key distribution scheme, each participant P_i reads the broadcast list and right-multiplies

$\mathsf{B}[i]$ by its own private matrix. For each P_i, we have

$$\mathsf{B}[i] \cdot \mathbf{P}_i = \mathsf{R}[i] \cdot \mathbf{P}_{N-1} \cdot \mathbf{P}_i \tag{34}$$

$$= \mathbf{v} \left(\prod_{k=0, k \neq i}^{N-2} \mathbf{P}_k \right) \cdot \mathbf{P}_{N-1} \cdot \mathbf{P}_i \tag{35}$$

$$= \mathbf{v} \left(\prod_{k=0, k \neq i}^{N-1} \mathbf{P}_k \right) \cdot \mathbf{P}_i = \mathbf{v} \prod_{k=0}^{N-1} \mathbf{P}_k, \tag{36}$$

the last equality in (36) originates from the commutativity of the matrices' product. We present the summary of this scheme in Table VI and its simulation procedure in Algorithm 4.

Setup for public parameters
The public parameters are identical to those described for the two-party protocol.
Generation of the private matrices
The procedure for generating the private matrices is identical to the procedure described in Table III.
Upflow round
Initialization for the upflow value
P_0 computes $\mathbf{v}\mathbf{P}_0$ and sends this value to P_1
For all $1 \leq i \leq N-2$,
(i) P_i receives the vector $\mathbf{v} \prod_{k=0}^{i-1} \mathbf{P}_k$ from P_{i-1}
(ii) P_i right-multiplies the vector $\mathbf{v} \prod_{k=0}^{i-1} \mathbf{P}_k$ by $\mathbf{P}_i$
(iii) P_i sends $\mathbf{v} \prod_{k=0}^{i} \mathbf{P}_k$ to P_{i+1}
Mutual key retrieval for P_{N-1}
P_{N-1} right-multiplies the vector from P_{N-2} by $\mathbf{P}_{N-1}$, that is: $\mathbf{v}\left(\prod_{k=0}^{N-2} \mathbf{P}_k\right) \mathbf{P}_{N-1} = \mathbf{v}\left(\prod_{k=0}^{N-1} \mathbf{P}_k\right)$, the result is the mutual secret vector
First Broadcast round
P_{N-2} broadcasts the value $\mathbf{v} \prod_{k=0}^{N-2} \mathbf{P}_k$ to all participant P_i with $0 \leq i \leq N-3$
Response round
For all $0 \leq i \leq N-2$,
(i) P_i computes $\mathbf{P}_i^{-1}$ (the inverse of its private matrix)
(ii) P_i calculates $\mathbf{v}\left(\prod_{k=0}^{N-2} \mathbf{P}_k\right) \mathbf{P}_i^{-1} = \mathbf{v} \prod_{k=0, k \neq i}^{N-2} \mathbf{P}_k$
(iii) P_i defines $\mathsf{R}[i] = \mathbf{v} \prod_{k=0, k \neq i}^{N-2} \mathbf{P}_k$ and sends $\mathsf{R}[i]$ to P_{N-1}
Second Broadcast round
(i) P_{N-1} constructs the broadcast list B of length $N-1$ where $\mathsf{B}[j] = \mathsf{R}[j] \cdot \mathbf{P}_{N-1}$ for $0 \leq j \leq N-2$
(ii) P_{N-1} broadcasts the list B to all P_i with $0 \leq i \leq N-2$
Mutual key retrieval for P_i $(0 \leq i \leq N-2)$
P_i reads $\mathsf{B}[i]$ and calculates $\mathsf{B}[i] \cdot \mathbf{P}_i$ to retrieve the mutual secret vector

TABLE VI: Megrelishvili group key distribution with upflow-broadcast-response rounds.

We state the correctness of this key distribution scheme in Theorem 2.

Theorem 2 *Each of the participants in Megrelishvili key distribution scheme described in Table VI receives an identical vector as its mutual key at the end of the second broadcast round.*

Proof: As early as the upflow round is over, P_{N-1} can retrieve the mutual key by right-multiplying the upflow vector from P_{N-2} by $\mathbf{P}_{N-1}$, we have $\mathbf{v}\left(\prod_{k=0}^{N-2} \mathbf{P}_k\right) \mathbf{P}_{N-1} = \mathbf{v} \prod_{k=0}^{N-1} \mathbf{P}_k$. Each participant P_i with $0 \leq i \leq N-2$ can retrieve the mutual key by right-multiplying the entry $\mathsf{B}[i]$ in the broadcast list by $\mathbf{P}_i$. Recall that from (34), (35), and (36), we have $\mathsf{B}[i] \cdot \mathbf{P}_i = \mathsf{R}[i] \cdot \mathbf{P}_{N-1} \cdot \mathbf{P}_i = \mathbf{v}\left(\prod_{k=0, k \neq i}^{N-2} \mathbf{P}_k\right) \cdot \mathbf{P}_{N-1} \cdot \mathbf{P}_i = \mathbf{v} \prod_{k=0}^{N-1} \mathbf{P}_k$. $\square$

Like the upflow-broadcast scheme in Section IV-C, the Megrelishvili key distribution scheme with upflow-broadcast-response rounds enables almost all participants to retrieve the mutual key simultaneously. Furthermore, we shall show that this protocol uses fewer right-multiplications (vector-matrix

Algorithm 4 A procedure for simulating the Megrelishvili key distribution using upflow-broadcast-response rounds.

```
Require: Public parameters as explained in Table VI and an integer $N \geq 3$ which denotes the number of group members.
 1: for $i \leftarrow 0$ to $N-1$ do                                   // $N$ group members
 2:     $\mathbf{P}_i \leftarrow \mathbf{M}^{\alpha_i}$              // private matrices generation for $N$ group members
 3: end for
 4: $\mathbf{u} \leftarrow \mathbf{v}\mathbf{P}_0$                    // initialization of the upflow vector
 5: $P_0$ transmits $\mathbf{u}$ to $P_1$
 6: for $i \leftarrow 1$ to $N-2$ do                                   // upflow round
 7:     $\mathbf{u} \leftarrow \mathbf{u}\mathbf{P}_i$               // right-multiplication by $\mathbf{P}_i$
 8:     $P_i$ sends $\mathbf{u}$ to $P_{i+1}$
 9: end for
10: key for $P_{N-1}$ is $\mathbf{u}\mathbf{P}_{N-1}$                  // key retrieval for $P_{N-1}$
11: $P_{N-2}$ broadcasts $\mathbf{u}$ to all participant $P_i$ with $0 \leq i \leq N-3$     // first broadcast round
12: for all $i \in [0, N-2]$ do                                        // response round
13:     $P_i$ calculates $\mathbf{P}_i^{-1}$ and $\mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\mathbf{P}_i^{-1} = \mathbf{v}\prod_{k=0,k\neq i}^{N-2}\mathbf{P}_k$
14:     $P_i$ defines $\mathsf{R}[i] \leftarrow \mathbf{v}\prod_{k=0,k\neq i}^{N-2}\mathbf{P}_k$
15:     $P_i$ sends $\mathsf{R}[i]$ to $P_{N-1}$
16: end for
17: for $i \leftarrow 0$ to $N-2$ do                                   // broadcast list construction by $P_{N-1}$
18:     $\mathsf{B}[i] \leftarrow \mathsf{R}[i] \cdot \mathbf{P}_{N-1}$
19: end for
20: $P_{N-1}$ broadcasts $\mathsf{B}$ to all participant $P_i$ with $0 \leq i \leq N-2$      // second broadcast round
21: for all $i \in [0, N-2]$ do
22:     key for $P_i$ is $\mathsf{B}[i] \cdot \mathbf{P}_i$           // key retrieval for $P_i$ $(0 \leq i \leq N-2)$
23: end for
Ensure: Each participant has an identical secret key.
```

multiplications) than the previous ones. From Table VI, we know that each participant P_i with $0 \leq i \leq N-2$ performs one right-multiplication during the upflow stage. This means that there are $N-1$ right-multiplications overall during this round. After the first broadcast round, each participant P_i with $0 \leq i \leq N-2$ performs one right-multiplication and sends the resulting value to P_{N-1}. The total number of right-multiplications in this stage is $N-1$. For constructing the broadcast list B, participant P_{N-1} collects the vectors from all other members and carries out $N-1$ right-multiplications. Each participant (including P_{N-1}) then performs one right-multiplication to obtain the mutual secret key to conclude the protocol. This implies that the key retrieval in the entire protocol requires N right-multiplications. From this analysis, the Megrelishvili key distribution scheme with upflow-broadcast-response rounds in Table VI requires

$$(N-1) + (N-1) + (N-1) + N = 4N - 3, \tag{37}$$

right-multiplications, which is $\left(N^2 - 11N + 8\right)/2$ operations fewer than (17) and (32). Thus, the number of right-multiplications in this scheme is linear with respect to the group size. By inspection, we deduce some significant characteristics of Megrelishvili protocol as follows:

1) total rounds: 4 (upflow, first broadcast, response, and second broadcast)
2) total messages transmitted per member: 2 for P_i with $0 \leq i \leq N-3$ (each contains one vector); 3 for P_{N-2} (each contains one vector); 1 for P_{N-1} (a list of $N-1$ vectors)
3) total messages received per member: 2 for P_0 and P_{N-2} (P_0 gets one vector from P_{N-2} and a list of $N-1$ vectors from P_{N-1}, P_{N-2} gets one vector from P_{N-3} and a list of $N-1$ vectors from P_{N-1}); 3 for P_i with $1 \leq i \leq N-3$ (one vector from P_{i-1}, one vector from P_{N-2}, and a list of $N-1$ vectors from P_{N-1}); N for P_{N-1} (each contains one vector)
4) total right-multiplications (vector-matrix multiplications) per member: 3 for P_i with $0 \leq i \leq N-2$

(one during the upflow round, one during the response round, and one for the mutual key retrieval), N for P_{N-1}.

5) total messages in the entire protocol: $2N$ with varying size from one to $N-1$ vectors
6) total right-multiplications in the entire protocol: $4N-3$.

V. Protocols for Group Membership Modification

All schemes in Section IV assume that the members of the group are determined prior to the execution of the protocols. However, sometimes it is necessary to insert a new or delete an existing participant in the protocols after the key distribution is completed. Pragmatically, it is enviable to perform so without having to re-execute the process all over again. In this section, we concisely propose the procedures for inserting a new and removing an existing group member for the upflow-broadcast scheme and the upflow-broadcast-response scheme. We choose these schemes due to their efficient nature in the key distribution process. Our procedures are inspired by the similar group membership alteration protocols explained in [5].

A. Protocols for a New Participant Insertion

Suppose initially there are N participants labeled as $P_0, P_1, \ldots, P_{N-1}$ who have completed a key distribution using upflow-broadcast scheme. The formation of a new mutual key for $N+1$ participants which consist of the original N members and a new member labeled as P_N needs to satisfy two specifications. First, the group does not need to re-run the key distribution anew. Second, the previous mutual key for N group members should remain secret from outsiders as well as P_N. In the upflow-broadcast scheme, these intentions can be achieved using the following protocols:

1) We assume that P_{N-1} saves the contents of the upflow list U from P_{N-2} as in (25).
2) To make the previous group key remains secret, P_{N-1} chooses a new integer $\hat{\alpha}_{N-1} \neq \alpha_{N-1}$ and generates $\hat{\mathbf{P}}_{N-1} = \mathbf{M}^{\hat{\alpha}_{N-1}}$. Additionally, P_{N-1} also needs to ensure that $\hat{\mathbf{P}}_{N-1} \neq \mathbf{P}_{N-1}$.
3) Participant P_{N-1} then constructs the new upflow list by right-multiplying each of the entries in U by $\hat{\mathbf{P}}_{N-1}$, the resulting updated list is

$$\mathsf{U} = \left[\mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}, \mathbf{v}\left(\prod_{k=0,k\neq N-2}^{N-2}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}, \ldots, \mathbf{v}\left(\prod_{k=0,k\neq 0}^{N-2}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}\right], \tag{38}$$

and P_{N-1} subsequently sends this list P_N.

4) Next, P_N chooses an integer α_N, generates $\mathbf{P}_N = \mathbf{M}^{\alpha_N}$, and computes
   a) $\mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}\mathbf{P}_N = \mathbf{v}\left(\prod_{k=0,k\neq N-1}^{N}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}$ for the mutual key,
   b) $\mathsf{B} = \left[\mathbf{v}\left(\prod_{k=0,k\neq N-1}^{N}\mathbf{P}_k\right), \mathbf{v}\left(\prod_{k=0,k\neq N-1,k\neq N-2}^{N}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}, \ldots, \mathbf{v}\left(\prod_{k=0,k\neq N-1,k\neq 0}^{N}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}\right]$ for the broadcast list.
5) After receiving the broadcast message from P_N, each participant P_i with $0 \leq i \leq N-1$ then computes the mutual key by right-multiplying $\mathsf{B}[N-1-i]$ by its own private matrix.

Member insertion in upflow-broadcast-response scheme is almost identical to that in upflow-broadcast scheme. Suppose there are N initial participants labeled as $P_0, P_1, \ldots, P_{N-1}$ and a new participant labeled as P_N. The protocol for inserting a new member has to comply with the efficiency and security aspects as described previously for the upflow-broadcast scheme. These intentions can be accomplished in the following steps:

1) We assume that P_{N-1} keeps the message from P_{N-2} (that is, the vector $\mathbf{v}\prod_{k=0}^{N-2}\mathbf{P}_k$) and the response list R where $\mathsf{R}[i] = \mathbf{v}\prod_{k=0,k\neq i}^{N-2}\mathbf{P}_k$ as described in Table VI.
2) To make the previous group key remains secret, P_{N-1} chooses a new integer $\hat{\alpha}_{N-1} \neq \alpha_{N-1}$ and generates $\hat{\mathbf{P}}_{N-1} = \mathbf{M}^{\hat{\alpha}_{N-1}}$. Additionally, P_{N-1} also needs to ensure that $\hat{\mathbf{P}}_{N-1} \neq \mathbf{P}_{N-1}$.
3) Subsequently, P_{N-1} performs the following steps:

a) Participant P_{N-1} right-multiplies the broadcast message from P_{N-2} by its new private matrix $\hat{\mathbf{P}}_{N-1}$ and sends the resulting value to P_N. Accordingly, P_N has the vector $\mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}$.
b) Participant P_{N-1} right-multiplies each of the elements in the response list R by $\hat{\mathbf{P}}_{N-1}$ and appends the vector $\mathbf{v}\prod_{k=0}^{N-2}\mathbf{P}_k$ to the resulting list. The outcome is denoted as the list R'. Afterward P_{N-1} sends R' to P_N.

4) We assume that participant P_N chooses an integer α_N and uses $\mathbf{P}_N = \mathbf{M}^{\alpha_N}$ as its private matrix.
5) After receiving the list R' from P_{N-1}, participant P_N right-multiplies each vector in R' by $\mathbf{P}_N$ and defines the resulting list as the broadcast list B. This list is then broadcast to all remaining group members.
6) For the key retrieval, P_N simply right-multiplies the vector $\mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}$ by $\mathbf{P}_N$ while each of the other group members computes the mutual key by right-multiplying the vector $\mathsf{B}[i]$ by its own private matrix (i.e., participant P_i with $0 \le i \le N-2$ computes $\mathsf{B}[i]\cdot\mathbf{P}_i$ while participant P_{N-1} computes $\mathsf{B}[N-1]\cdot\hat{\mathbf{P}}_{N-1}$).

B. *Protocols for an Existing Participant Removal*

We consider a group of N members $P_0, P_1, \ldots, P_{N-1}$ who have completed a key distribution using upflow-broadcast scheme. Suppose the group wants to remove a member P_r for some $0 \le r \le N-1$. The formation of a new mutual key for $N-1$ participants needs to fulfil two properties. First, the group does not need to re-run the key distribution all over again. Second, the new mutual key for the initial N members should remain secret from outsiders as well as P_r. In the following member removal protocol for upflow-broadcast scheme, we assume that $r \neq N-1$. The key agreement steps are as follows:

1) Participant P_{N-1} is assumed to have save the upflow list U of length N from P_{N-2} as in (25).
2) To ensure that the initial group key remains secret, P_{N-1} chooses a new integer $\hat{\alpha}_{N-1} \neq \alpha_{N-1}$ and generates $\hat{\mathbf{P}}_{N-1} = \mathbf{M}^{\hat{\alpha}_{N-1}}$. Additionally, P_{N-1} also needs to ensure that $\hat{\mathbf{P}}_{N-1} \neq \mathbf{P}_{N-1}$.
3) The key retrieval for P_{N-1} is performed by right-multiplying $\mathsf{U}[0]$ by $\hat{\mathbf{P}}_{N-1}$. The new mutual key is $\mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}$.
4) In the case that P_r (for some $0 \le r \le N-2$) is detached from the group, P_{N-1} needs to ensure that only P_i with $0 \le i \le N-1$ and $i \neq r$ can retrieve the new mutual key. Thus, P_{N-1} defines a broadcast list B of length $N-1$ as follows:

$$\mathsf{B}[i] = \begin{cases} \mathsf{U}[i+1]\cdot\hat{\mathbf{P}}_{N-1}, & \text{for all } 0 \le i \le N-2 \text{ with } i \neq N-2-r \\ \mathbf{v} \text{ (the public vector)}, & \text{otherwise.} \end{cases} \tag{39}$$

Subsequently, P_{N-1} broadcast this list to all other group members. Observe that (39) is analogous to (26).
5) The key retrieval for P_i with $0 \le i \le N-2$ and $i \neq r$ is performed by calculating $\mathsf{B}[N-2-i]\cdot\mathbf{P}_i$. In this case, we have

$$\mathsf{B}[N-2-i]\cdot\mathbf{P}_i = \mathsf{U}[N-1-i]\cdot\hat{\mathbf{P}}_{N-1}\cdot\mathbf{P}_i \tag{40}$$

$$= \mathbf{v}\left(\prod_{k=0,k\neq i}^{N-2}\mathbf{P}_k\right)\cdot\hat{\mathbf{P}}_{N-1}\cdot\mathbf{P}_i = \mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\cdot\hat{\mathbf{P}}_{N-1}. \tag{41}$$

This procedure ensures that the removed member P_r unable to compute the new mutual key because the vector $\mathbf{v}\left(\prod_{k=0,k\neq r}^{N-2}\mathbf{P}_k\right)\cdot\hat{\mathbf{P}}_{N-1}$ is missing from the list. In particular, the operation $\mathsf{B}[N-2-r]\cdot\mathbf{P}_r$ yields $\mathbf{v}\mathbf{P}_r$. In the event that P_{N-1} is removed from the group, P_{N-2} assumes the role as the last participant as explained earlier.

Member removal in upflow-broadcast-response scheme is almost similar to that in upflow-broadcast scheme. Suppose there are N initial participants labeled as $P_0, P_1, \ldots, P_{N-1}$ and the group wants to remove P_r for some $0 \le r \le N-1$. The protocol for removing a member needs to fulfil the efficiency and security requirements as described previously for the upflow-broadcast scheme. If $r \neq N-1$, these purposes can be attained using the following steps:

1) Participant P_{N-1} is assumed to retain the message from P_{N-2} (that is, the vector $\mathbf{v}\prod_{k=0}^{N-2}\mathbf{P}_k$) and the response list R where $\mathsf{R}[i] = \mathbf{v}\prod_{k=0,k\neq i}^{N-2}\mathbf{P}_k$ as described in Table VI.
2) To ensure that the initial group key remains secret, P_{N-1} chooses a new integer $\hat{\alpha}_{N-1} \neq \alpha_{N-1}$ and generates $\hat{\mathbf{P}}_{N-1} = \mathbf{M}^{\hat{\alpha}_{N-1}}$. Additionally, P_{N-1} also needs to ensure that $\hat{\mathbf{P}}_{N-1} \neq \mathbf{P}_{N-1}$.
3) The key retrieval for P_{N-1} is performed by right-multiplying the message from P_{N-2} by $\hat{\mathbf{P}}_{N-1}$. The new mutual key is $\mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\hat{\mathbf{P}}_{N-1}$.
4) In the case that P_r (for some $0 \leq r \leq N-2$) is removed from the group, P_{N-1} needs to ensure that only P_i with $0 \leq i \leq N-1$ and $i \neq r$ can retrieve the new mutual key. Thus, P_{N-1} defines a broadcast list B of length $N-1$ as follows:

$$\mathsf{B}[i] = \begin{cases} \mathsf{R}[i]\cdot\hat{\mathbf{P}}_{N-1}, & \text{for all } 0 \leq i \leq N-2 \text{ with } i \neq r \\ \mathbf{v}\ \text{(the public vector)}, & \text{otherwise.} \end{cases} \tag{42}$$

Afterward, P_{N-1} broadcast this list to all other group members. Observe that (39) is analogous to the definition of the ordinary broadcast list in Table VI.
5) The key retrieval for P_i with $0 \leq i \leq N-2$ and $i \neq r$ is performed by calculating $\mathsf{B}[i]\cdot\mathbf{P}_i$. In this case, we have

$$\mathsf{B}[i]\cdot\mathbf{P}_i = \mathsf{R}[i]\cdot\hat{\mathbf{P}}_{N-1}\cdot\mathbf{P}_i \tag{43}$$

$$= \mathbf{v}\left(\prod_{k=0,k\neq i}^{N-2}\mathbf{P}_k\right)\cdot\hat{\mathbf{P}}_{N-1}\cdot\mathbf{P}_i = \mathbf{v}\left(\prod_{k=0}^{N-2}\mathbf{P}_k\right)\cdot\hat{\mathbf{P}}_{N-1}. \tag{44}$$

This procedure ensures that the removed member P_r unable to compute the new mutual key because the vector $\mathbf{v}\left(\prod_{k=0,k\neq r}^{N-2}\mathbf{P}_k\right)\cdot\hat{\mathbf{P}}_{N-1}$ is missing from the list. In particular, we have $\mathsf{B}[r]\cdot\mathbf{P}_r = \mathbf{v}\mathbf{P}_r$. In the event that P_{N-1} is removed from the group, P_{N-2} assumes the role as the last participant as explained earlier.

VI. ELEMENTARY THEORETICAL SECURITY ANALYSIS

One important security requirement for the key distribution protocols in Section IV and Section V is the secrecy of the mutual group key created. The protocols must ensure that no outsider can recover the mutual key easily. We notice that in a group of N participants $P_0, P_1, \ldots, P_{N-1}$, the mutual key is $\mathbf{v}\prod_{k=0}^{N-1}\mathbf{P}_k$. Let $\mathcal{I} = \{0, 1, \ldots, N-1\}$, then the mutual key can be rewritten as $\mathbf{v}\prod_{k\in\mathcal{I}}\mathbf{P}_k$ due to the commutativity of the private matrices. By observation, any message sent during the transmission in our key distribution protocols contains at least one vector of the form $\mathbf{v}\prod_{k\in\mathcal{J}}\mathbf{P}_k$ for some $\mathcal{J} \subset \mathcal{I}$. An eavesdropper (Eve) may recover the mutual key if she can reconstruct the value $\mathbf{v}\prod_{k\in\mathcal{I}}\mathbf{P}_k$ from several vectors of the form $\mathbf{v}\prod_{k\in\mathcal{J}}\mathbf{P}_k$ with $\mathcal{J} \subset \mathcal{I}$.

More formally, suppose $\{\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_m\}$ denotes a collection of m non-empty subset of $\mathcal{I}$, that is $\mathcal{J}_i \subset \mathcal{I}$ for all $1 \leq i \leq m$. We assume that Eve intercepts the transmission and accordingly owns m vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$, and each $\mathbf{w}_i$ satisfies $\mathbf{w}_i = \mathbf{v}\prod_{k\in\mathcal{J}_i}\mathbf{P}_k$ with $1 \leq i \leq m$. To recover the mutual key, she must be able to calculate the value of $\mathbf{v}\prod_{k\in\mathcal{I}}\mathbf{P}_k$ using only m vectors $\mathbf{w}_i$ where $1 \leq i \leq m$. In addition, it is necessary that $\bigcup_{i=1}^{m}\mathcal{J}_i = \mathcal{I}$, otherwise the mutual key cannot be obtained. However, this condition alone does not guarantee that the mutual key can be acquired easily. One main problem is because the matrices $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{N-1}$ are private and unknown to the outsider.

To overcome the problem, Eve needs to consider all public parameters before she try to recover the key. The matrices $\mathbf{P}_i$ $(0 \leq i \leq N-1)$ are private, but fortunately for Eve, each of these matrices can be expressed as $\mathbf{P}_i = \mathbf{M}^{\alpha_i}$ where $\mathbf{M}$ is a public matrix and α_i is a secret integer. Using this relationship, we have

$$\mathbf{v}\prod_{k\in\mathcal{J}}\mathbf{P}_k = \mathbf{v}\prod_{k\in\mathcal{J}}\mathbf{M}^{\alpha_k} = \mathbf{v}\mathbf{M}^{\sum_{k\in\mathcal{J}}\alpha_k}, \tag{45}$$

which possibly makes the mutual secret key recovery less complicated. If we assume that Eve gets m vectors of the form $\mathbf{w}_i = \mathbf{v}\prod_{k\in\mathcal{J}_i}\mathbf{P}_k$ with $1 \leq i \leq m$, then by (45), each $\mathbf{w}_i$ can be expressed as $\mathbf{vM}^{\sum_{k\in\mathcal{J}_i}\alpha_k}$. For brevity, we write $\sum_{k\in\mathcal{J}_i}\alpha_k = x_i$. From theoretical perspective, the security of Megrelishvili key distribution protocols in Section IV and Section V relates to the following problem.

Definition 1 (Multi-party Megrelishvili Shared Key Problem [1]) Suppose $\{\mathcal{J}_1,\mathcal{J}_2,\ldots,\mathcal{J}_m\}$ denotes a collection of m non-empty proper subset of $\mathcal{I}=\{0,1,\ldots,N-1\}$ and for each $\mathcal{J}_i$ $(1\le i\le m)$ we define $x_i=\sum_{k\in\mathcal{J}_i}\alpha_k$. The multi-party Megrelishvili shared key problem (MMSKP) is the problem of determining the value $\mathbf{vM}^x$ where $x=\sum_{k\in\mathcal{I}}\alpha_k$ from the know set $\{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_m\}$ where each $\mathbf{w}_i=\mathbf{vM}^{x_i}$.

MMSKP is the problem of determining the mutual group key in Megrelishvili key distribution scheme using known vectors obtained during the public (non-secure) messages transmission. This problem was first discussed in [1]. It is obvious that in order to solve MMSKP, we must have $\bigcup_{i=1}^m\mathcal{J}_i=\mathcal{I}$. However, this condition alone is not always sufficient (see [1, Example 3] for more detailed explanation). One palpable initial approach to solve MMSKP in Definition 1 is by solving m instances of MVMP. That is, Eve solves $\mathbf{w}_i=\mathbf{vM}^{x_i}$ for the value of x_i for all $1\le i\le m$. Subsequently, Eve tries to express x as a real-valued function of $x_1,x_2,\ldots,x_m$. If we have $x=f(x_1,x_2,\ldots,x_m)$ for some function f, then the secret key is equivalent to $\mathbf{vM}^{f(x_1,x_2,\ldots,x_m)}$. The problem of recovering the mutual key is now reduce to finding an appropriate function f of m variables $x_1,x_2,\ldots,x_m$ such that $f(x_1,x_2,\ldots,x_m)=x$.

We propose an elementary linear function that can be used to express x as a function of $x_1,x_2,\ldots,x_m$. More precisely, our function f has the form

$$f(x_1,x_2,\ldots,x_m)=k_1x_1+k_2x_2+\cdots+k_mx_m, \tag{46}$$

for some real-valued scalars $k_1,k_2,\ldots,k_m$. Thus, the secret exponent x can be recovered if x can be expressed as a linear combination of $x_1,x_2,\ldots,x_m$, where each x_i is the secret exponent that corresponds to the equation $\mathbf{w}_i=\mathbf{vM}^{x_i}$ for some known vector $\mathbf{w}_i$. Observe that each x_j with $1\le j\le m$ can be expressed as

$$x_j=c_{0,j}\alpha_0+c_{1,j}\alpha_1+\cdots+c_{N-1,j}\alpha_{N-1}=\sum_{i=0}^{N-1}c_{i,j}\alpha_i \tag{47}$$

where $c_{i,j}\in\{0,1\}$ and some (but not all) $c_{i,j}$ equals to 1. Consequently, any subset $\mathcal{J}_j$ of the non-empty subset collection $\{\mathcal{J}_1,\mathcal{J}_2,\ldots,\mathcal{J}_m\}$ corresponds to the secret exponent $x_j=\sum_{i=0}^{N-1}c_{i,j}\alpha_i$. Furthermore, the non-empty subset collection $\{\mathcal{J}_1,\mathcal{J}_2,\ldots,\mathcal{J}_m\}$ corresponds to the set $\mathcal{E}=\{x_1,x_2,\ldots,x_m\}$ of secret exponents extracted from m MVMP instances. We now have the following important results.

Lemma 2 *Let $\mathcal{E}=\{x_1,x_2,\ldots,x_m\}=\left\{\sum_{i=0}^{N-1}c_{i,j}\alpha_i : c_{i,j}\in\{0,1\}\wedge(1\le j\le m)\right\}$ be the set of m secret exponents extracted from the transmission among N participants $P_0,P_1,\ldots,P_{N-1}$. The mutual group key $\mathbf{vM}^x$ where $x=\sum_{i=0}^{N-1}\alpha_i$ is recoverable from the set $\mathcal{E}$ if the equation*

$$\sum_{j=1}^m k_j\left(\sum_{i=0}^{N-1}c_{i,j}\alpha_i\right)=\sum_{i=0}^{N-1}\alpha_i \tag{48}$$

is solvable for k_j $(1\le j\le m)$. In other words, the mutual key is recoverable if we can find the value k_j for $1\le j\le m$ that makes (48) is satisfied.

Proof: The mutual key is recoverable if we can find scalar k_j $(1\le j\le m)$ such that $k_1x_1+k_2x_2+\cdots+k_mx_m=x$, observe that

$$\begin{aligned}k_1x_1+k_2x_2+\cdots+k_mx_m&=x\\ \sum_{j=1}^m k_jx_j&=\sum_{i=0}^{N-1}\alpha_i\\ \sum_{j=1}^m k_j\left(\sum_{i=0}^{N-1}c_{i,j}\alpha_i\right)&=\sum_{i=0}^{N-1}\alpha_i \text{ (by 47).}\end{aligned}$$

□

If we expand $k_j\left(\sum_{i=0}^{N-1} c_{i,j}\alpha_i\right)$, then we have

$$k_j\left(\sum_{i=0}^{N-1} c_{i,j}\alpha_i\right) = k_j\left(c_{0,j}\alpha_0 + c_{1,j}\alpha_1 + \cdots + c_{N-1,j}\alpha_{N-1}\right) \tag{49}$$

$$= k_j c_{0,j}\alpha_0 + k_j c_{1,j}\alpha_1 + \cdots + k_j c_{N-1,j}\alpha_{N-1}. \tag{50}$$

By substituting the expansion (50) to (48), we get

$$\begin{aligned}\sum_{j=1}^{m} k_j\left(\sum_{i=0}^{N-1} c_{i,j}\alpha_i\right) &= \sum_{j=1}^{m}\left(k_j c_{0,j}\alpha_0 + k_j c_{1,j}\alpha_1 + \cdots + k_j c_{N-1,j}\alpha_{N-1}\right)\\ &= k_1 c_{0,1}\alpha_0 + k_1 c_{1,1}\alpha_1 + \cdots + k_1 c_{N-1,1}\alpha_{N-1}\\ &+ k_2 c_{0,2}\alpha_0 + k_2 c_{1,2}\alpha_1 + \cdots + k_2 c_{N-1,2}\alpha_{N-1}\\ &+ \cdots\\ &+ k_m c_{0,m}\alpha_0 + k_m c_{1,m}\alpha_1 + \cdots + k_m c_{N-1,m}\alpha_{N-1},\end{aligned}$$

which can be expressed as

$$\begin{aligned}\sum_{j=1}^{m} k_j\left(\sum_{i=0}^{N-1} c_{i,j}\alpha_i\right) &= \alpha_0\left(\sum_{j=1}^{m} k_j c_{0,j}\right) + \alpha_1\left(\sum_{j=1}^{m} k_j c_{1,j}\right) + \cdots + \alpha_{N-1}\left(\sum_{j=1}^{m} k_j c_{N-1,j}\right)\\ &= \sum_{i=0}^{N-1}\left(\sum_{j=1}^{m} k_j c_{i,j}\right)\alpha_i. \end{aligned}\tag{51}$$

By Lemma 2, the mutual key is recoverable if we have

$$\sum_{j=1}^{m} k_j\left(\sum_{i=0}^{N-1} c_{i,j}\alpha_i\right) = \sum_{i=0}^{N-1}\left(\sum_{j=1}^{m} k_j c_{i,j}\right)\alpha_i = \sum_{i=0}^{N-1}\alpha_i. \tag{52}$$

By matching the terms α_i for $0 \le i \le N-1$ in (52), we have

$$\sum_{j=1}^{m} k_j c_{i,j} = \sum_{j=1}^{m} c_{i,j} k_j = 1 \text{ for all } 0 \le i \le N-1. \tag{53}$$

The expression (53) can be expanded as follows

$$\begin{array}{ccccccccc} c_{0,1}k_1 & + & c_{0,2}k_2 & + & \cdots & + & c_{0,m}k_m & = & 0\\ c_{1,1}k_1 & + & c_{1,2}k_2 & + & \cdots & + & c_{1,m}k_m & = & 0\\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots\\ c_{N-1,1}k_1 & + & c_{N-1,2}k_2 & + & \cdots & + & c_{N-1,m}k_m & = & 0 \end{array} \tag{54}$$

The system of equations in (54) can be expressed in matrix equation form:

$$\begin{bmatrix} c_{0,1} & c_{0,2} & \cdots & c_{0,m}\\ c_{1,1} & c_{1,2} & \cdots & c_{1,m}\\ \vdots & \vdots & \ddots & \vdots\\ c_{N-1,1} & c_{N-1,2} & \cdots & c_{N-1,m} \end{bmatrix}\begin{bmatrix} k_1\\ k_2\\ \vdots\\ k_m\end{bmatrix} = \begin{bmatrix} 1\\ 1\\ \vdots\\ 1\end{bmatrix} \tag{55}$$

$$\mathbf{A}\mathbf{k} = \mathbf{1}, \tag{56}$$

where $\mathbf{A} = [a_{i,j}]$ is an $N \times m$ matrix over $\{0,1\}$ with $a_{i,j} = c_{i-1,j}$ for all $1 \le i \le N$ and $1 \le j \le m$, $\mathbf{k}$ is a column vector of unknowns of size m, and $\mathbf{1}$ is a column vector of N ones. Therefore, the solvability of the matrix equation (56) implies the recoverability of the mutual secret key. This stipulation leads to a sufficient condition for the recoverability of the mutual secret key in connection with the collection of secret exponents excerpted from the transmission.

Theorem 3 *Assume that the eavesdropper intercepts the message and excerpts the set*

$$\mathcal{E} = \{x_1, x_2, \ldots, x_m\} = \left\{ \sum_{i=0}^{N-1} c_{i,j}\alpha_i : c_{i,j} \in \{0,1\} \wedge (1 \leq j \leq m) \right\}$$

of m secret exponents from a key distribution scheme among N participants $P_0, P_1, \ldots, P_{N-1}$. Suppose $\mathbf{A} = [a_{i,j}]$ is an $N \times m$ matrix over $\{0,1\}$ with $a_{i,j} = c_{i-1,j}$ for all $1 \leq i \leq N$ and $1 \leq j \leq m$. Let $\mathbf{A}'$ be the $N \times (m+1)$ augmented matrix of the matrix equation (56). If the reduced row echelon form of $\mathbf{A}'$ contains no row of the form $[\mathbf{0}_m \ *]$ where $\mathbf{0}_m$ denotes a submatrix of size $1 \times m$ and $*$ denotes any nonzero value, then the mutual secret key is recoverable from the set $\mathcal{E}$. Additionally, the sum of all secret exponents of N participant can be expressed as a linear combination of the elements in $\mathcal{E}$.

Proof: Suppose $\mathbf{A} = [a_{i.j}]$ is an $N \times m$ matrix over $\{0,1\}$ with $a_{i,j} = c_{i-1,j}$ for all $1 \leq i \leq N$ and $1 \leq j \leq m$. From elementary linear algebra, the matrix equation $\mathbf{Ak} = \mathbf{1}$ in (56) has a solution for $\mathbf{k}$ if and only if the reduced row echelon form of the augmented matrix $\mathbf{A}' = [\mathbf{A} \ \mathbf{1}]$ has no row of the form $[\mathbf{0}_m \ *]$ where $\mathbf{0}_m$ is a submatrix of size $1 \times m$ and $*$ is any nonzero value. Accordingly, the condition that $\mathbf{A}'$ has no row of the form $[\mathbf{0}_m \ *]$ implies that there are scalars $k_1, k_2, \ldots, k_m$ that makes (48) is satisfied. By Lemma 2, the later condition implies that the sum of all secret exponents of N participants can be expressed as a linear combination of m secret exponents excerpted from the messages transmission. □

Theorem 3 provides a sufficient condition for recovering the mutual group key from several secret exponents excerpted from the messages transmission. In particular, this theorem states a sufficient condition for the sum of all private exponents $\alpha_0, \alpha_1, \ldots, \alpha_{N-1}$ of N participants $P_0, P_1, \ldots, P_{N-1}$ to be expressed as a linear combination of the secret exponents extracted from the messages during the public transmission. From the aforementioned analysis, we see that MMSKP can be reduced to several instances of MVMP, thus making MMSKP is not computationally harder than MVMP[4]. In addition, we conjecture that an eavesdropper should be able to solve MVMP in order to solve MMSKP. Thus far, the fastest known algorithm for solving MVMP still requires exponential number of scalar operations in terms of the vector space dimension used [14]. By this assumption, we argue that the Megrelishvili key distribution scheme is at least as secure as its two-party counterpart.

VII. CONCLUDING REMARKS

We have presented an extension of our previous work in [1] where we discuss the two first variations of Megrelishvili key distribution scheme and some of their elementary theoretical security analysis. In this article we introduce two different multi-party Megrelishvili protocols which are more efficient than those two first schemes. We propose two efficient Megrelishvili key distribution protocols, i.e.: the Megrelishvili key distribution with upflow-broadcast rounds and the Megrelishvili key distribution with upflow-broadcast-response rounds. These two schemes allow simultaneous key retrieval for almost all group members whilst maintaining the efficiency of the computational procedures involved. In addition, both schemes support the group membership alteration protocols. That is, the protocols enable the group to construct a new mutual secret key whenever a new member is added or an existing member is removed without re-executing the protocols all over again. The comparison of important characteristics for all Megrelishvili key distribution schemes described in Section IV is summarized in Table VII.

From Table VII, it can be derived that Megrelishvili key distribution scheme with upflow-broadcast-response rounds is more superior than the other three schemes in terms of computational efficiency. This protocol allows the group to agree on a mutual secret key using $\mathcal{O}(N)$ number of vector-matrix multiplications where N is the group size. Moreover, this scheme requires no *a priori* synchronization of the group members, yet it still supports simultaneous key retrieval for almost all participants (i.e., all but one participant). Another important feature of this key distribution procedure is the easiness for the implementation of membership alteration protocol.

[4]That is, the mutual secret vector can be obtained in polynomial number of scalar operations provided that appropriate secret exponents are extracted from the messages during the public transmission.

	Generic	U-D	U-B	U-B-R
total rounds (stages)	$N-1$	2	2	4
messages sent per member	$N-1$	2 or 1	1	2, 3, or 1
messages received per member	$N-1$	2 or 1	2 or 1	2, 3, or N
vector-matrix multiplications per member P_i	N	$i+2$ or N	$i+2$ or N	3 or N
total messages	$N(N-1)$	$2(N-1)$	N	$2N$
total vector-matrix multiplications	N^2	$\left(N^2+3N-2\right)/2$	$\left(N^2+3N-2\right)/2$	$4N-3$
a priori synchronization	Yes	No	No	No
uniform computation	Yes	No	No	No
simultaneous key retrieval for all/almost all members	Yes	No	Yes	Yes
efficient protocol for membership alteration	No	No	Yes	Yes

TABLE VII: Comparison of the generic protocol, upflow-downflow (U-D) scheme, upflow-broadcast (U-B) scheme, and upflow-broadcast-response (U-B-R) scheme.

One noticeable common characteristic of the key distribution with upflow-downflow rounds, upflow-broadcast rounds, and upflow-broadcast-response rounds is the non-necessity of the synchronization before the key exchange takes place. This characteristic also implies that the computational burden for each of the group members can be differ from one to another. Moreover, the computational and communication cost for each of the group members depend on its appearance in a linear order during the upflow round.

We have discussed some elementary security aspects of the Megrelishvili key distribution scheme. We base our investigation on multi-party Megrelishvili shared key problem (MMSKP) and we provide a mathematical relationship between MMSKP and MVMP. Specifically, we propose a method for solving MMSKP by way of solving several MVMP instances. In this method, we give a sufficient condition for recovering the mutual group key from several secret exponents excerpted from the messages transmission in Theorem 3. More precisely, we show that the mutual group key can be recovered efficiently whenever the sum of secret exponents of the participants can be expressed as a linear combination of the exponents extracted from the messages transmission. Nevertheless, we do not know yet whether this sufficient condition is also necessary in general approach of solving the MMSKP. Based on this analysis, we argue that our protocols are at least as secure as the original two-party key exchange procedure. However, a more rigorous theoretical research for the security of Megrelishvili key distributions still needs to be conducted.

Acknowledgment

The author would like to thank both reviewers for their feedbacks as well as the Students' Research Group for Cryptography at Computing Laboratory, Telkom University, Indonesia. This research is financially supported by the Directorate of Research and Community Service (*Direktorat Penelitian dan Pengabdian Masyarakat*, PPM), Telkom University, Indonesia.

References

[1] 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.

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

[3] 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.

[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] 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.

[6] A. Joux, “A one round protocol for tripartite Diffie–Hellman,” *Journal of Cryptology*, vol. 17, no. 4, pp. 263–276, 2004.

[7] D. Shanks, “Class number, a theory of factorization, and genera,” in *Proc. Symp. Pure Math*, vol. 20, 1971, pp. 415–440.

[8] S. Pohlig and M. Hellman, “An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance (Corresp.),” *IEEE Transactions on information Theory*, vol. 24, no. 1, pp. 106–110, 1978.

[9] J. M. Pollard, “Monte Carlo methods for index computation $(\bmod p)$,” *Mathematics of Computation*, vol. 32, no. 143, pp. 918–924, 1978.

[10] V. Shoup, “Lower bounds for discrete logarithms and related problems,” in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1997, pp. 256–266.

[11] 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.

[12] M. Arzaki, “Elementary algorithms analysis of Megrelishvili protocol,” *Indonesian Journal on Computing (Indo-JC)*, vol. 1, no. 1, pp. 11–23, 2016.

[13] M. Arzaki and B. A. Wahyudi, “On the construction of secure public parameters for Megrelishvili protocol,” in *2016 4th International Conference on Information and Communication Technology (ICoICT 2016)*, Bandung, Indonesia, May 2016.

[14] ——, “Collision algorithms for breaking Megrelishvili protocol: theory and numerical experiments,” in *2016 8th International Conference on Advanced Computer Science and Information Systems (ICACSIS 2016)*, 2016.

[15] I. Niven, “Fermat’s theorem for matrices,” *Duke Mathematical Journal*, vol. 15, no. 3, pp. 823–826, 1948.

[16] R. Megrelishvili and A. Sikharulidze, “New matrix sets generation and the cryptosystems,” in *Proceedings of the European Computing Conference and the Third International Conference on Computational Intelligence, Tbilisi, Georgia*, 2009, pp. 253–255.

[17] R. Megrelishvili, M. Chelidze, and G. Besiashvili, “Investigation of new matrix-key function for the public cryptosystems,” in *Proceedings of The Third International Conference, Problems of Cybernetics and Information*, vol. 1, 2010, pp. 6–8.

[18] R. Megrelishvili, “On the original one-way function and the new digital signature scheme,” *Applied Mathematics, Informatics and Mechanics (AMIM)*, vol. 16, no. 1, pp. 43–48, 2011.

[19] A. Beletsky, A. Beletsky, and R. Kandyba, “Matrix analogues of the Diffie-Hellman protocol,” in *ICTERI*, 2013, pp. 352–359.