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

IND. JOURNAL ON COMPUTING
VOL. 1, ISSUE. 1, MARCH 2016. PP. 11-24
DOI:10.21108/INDOJC.2016.1.1.52

Elementary Algorithms Analysis of Megrelishvili Protocol

Muhammad Arzaki

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

arzaki@telkomuniversity.ac.id

Abstract

This article presents an investigation of asymptotic time complexities of several algorithms related to Megrelishvili protocol. The analysis are carried out for the private keys computations and public exchange of values, public key constructions, as well as an elementary exhaustive search attack algorithm. We show that the complexities of these algorithms are higher than the complexities of elementary algorithms involved in the conventional Diffie-Hellman protocol (DHP) or its variant on elliptic curves (ECDHP). Furthermore, we introduce the notion of Megrelishvili vector-matrix problem as a linear algebraic problem that underlies the security of this protocol. Finally, we prove that under similar elementary exhaustive search attack algorithms, the number of scalar calculations required to solve this problem is greater than the number of scalar calculations required to solve the conventional discrete logarithm problem (DLP) and elliptic curves discrete logarithm problem (ECDLP). This condition intuitively implies that Megrelishvili protocol is "more secure" than DHP and ECDHP against exhaustive search attack.

Keywords: algorithms analysis, asymptotic complexities, Megrelishvili protocol.

Abstrak

Makalah ini menyajikan investigasi terhadap kompleksitas waktu asimtotik untuk beberapa algoritma yang berkaitan dengan protokol Megrelishvili. Analisis dilakukan pada komputasi untuk kunci rahasia dan pertukaran nilai-nilai publik, konstruksi kunci publik, serta algoritma serangan pencarian menyeluruh. Kami menunjukkan bahwa kompleksitas algoritma-algoritma ini lebih tinggi daripada kompleksitas algoritma-algoritma dasar yang terlibat dalam protokol Diffie-Hellman konvensional (DHP) atau variannya pada kurva eliptik (ECDHP). Selain itu, kami memperkenalkan gagasan masalah vektor-matriks Megrelishvili sebagai masalah aljabar linier yang mendasari keamanan protokol ini. Pada akhirnya, kami membuktikan bahwa dengan serangan pencarian menyeluruh yang serupa, banyaknya kalkulasi skalar yang diperlukan untuk memecahkan masalah ini lebih besar daripada banyaknya perhitungan skalar yang diperlukan dalam menyelesaikan masalah logaritma diskret konvensional (DLP) maupun masalah logaritma diskret pada kurva eliptik (ECDLP). Kondisi ini secara intuitif mengatakan bahwa protokol Megrelishvili "lebih aman "terhadap serangan pencarian menyeluruh dibandingkan dengan DHP maupun ECDHP.

Kata Kunci: analisis algoritma, kompleksitas asimtotik, protokol Megrelishvili.

I. INTRODUCTION

ONE important purpose of public key cryptosystems is to enable two or more parties to exchange confidential information via an open channel. This intention has been widely studied since the discovery and introduction of the Diffie-Hellman key exchange protocol [1]. This protocol has been broadly used as a building block for many new key exchange algorithms. The original protocol can be modified to work in arbitrary finite groups. The finite groups can be elliptic curve groups over finite

RECEIVED ON FEBRUARY 2016. ACCEPTED ON MARCH 2016

fields [2], [3], general linear groups (group of matrices) [4], [5], or symmetric groups [6]. Despite the variations of the finite groups used in each protocol, the security of these protocols relies on the similar discrete logarithm problems.

Megrelishvili protocol was first introduced by R. Megrelishvili, M. Chelidze, and K. Chelidze in [7]. Unlike the regular matrix-based Diffie-Hellman protocol, Megrelishvili key exchange algorithm combines matrix exponentiation and vector-matrix multiplication in its private keys generations. The combination of these two linear algebraic operations yields a key agreement whose security does not rely on the standard discrete logarithm problems for finite groups. Albeit its uniqueness, to our knowledge, this relatively new key exchange scheme has not been widely implemented. One purported reason is because the natural definition of a matrix multiplication requires at least $\Omega\left(n^2\right)$ steps of scalar operations [8]. This lower bound of the number of scalar operations leads to a conjecture that the computational costs to generate the keys in Megrelishvili protocol are greater than the computational costs required in the typical number theoretic-based key agreements (e.g. conventional Diffie-Hellman protocol). To answer this conjecture, we conduct a formal mathematical analysis for several elementary algorithms related to Megrelishvili protocol.

In this paper, we investigate the asymptotic time complexities for several elementary algorithms involved in Megrelishvili protocol. To our knowledge, the formal algorithms analysis concerning the running time of these algorithms has not been studied extensively. For this reason, the objective of this article is to provide formal mathematical analysis for these elementary algorithms. The analysis are carried out primarily for the private keys computations, public key construction, and exhaustive search attack algorithm.

II. MATHEMATICAL PRELIMINARIES

Throughout this paper, we denote a group G together with an operation $*$ in G by the pair $(G,*)$. All groups are written multiplicatively, unless otherwise specified. The order of G, denoted by $|G|$, is defined as the cardinality of G. All groups in this paper are assumed to have finite order, i.e. $|G|=n$ for some integer n. We define the order of $g\in G$, denoted by $o\left(g\right)$, as the least positive integer t satisfying $g^t=e$, where e is the identity element of G. It is a well-known fact in modern algebra that $o\left(g\right)$ divides $|G|$ [9], [10].

Megrelishvili protocol makes use of linear algebraic theory in its public key creation, private keys calculations, public exchange of values, and shared secret key calculations. The matrices and vector spaces used in this key agreement are defined over finite fields $\mathbb{F}_q$. We denote the set of all non zero elements of $\mathbb{F}_q$ by $\mathbb{F}_q^*$. For example, if $q=3$, we have $\mathbb{F}_3=\{0,1,2\}$ and $\mathbb{F}_3^*=\{1,2\}$. We represent an n-dimensional vector space over $\mathbb{F}_q$ by $\mathbb{F}_q^n$, the set of all $n\times n$ matrices over $\mathbb{F}_q$ by $M_n\left(q\right)$, and the set of all nonsingular $n\times n$ matrices over $\mathbb{F}_q$ by $GL_n\left(q\right)$. All vectors are denoted by right-pointing arrow notation, e.g. $\vec{v}$, while matrices are denoted by boldface upper-case letters. The vectors are considered and treated as row vectors, and therefore, the vector-matrix multiplication $\vec{v}\mathbf{M}$ is well-defined whenever $\vec{v}$ has n components and $\mathbf{M}$ is an $n\times n$ matrix.

Given a matrix $\mathbf{M}\in GL_n\left(q\right)$, the (multiplicative) order of $\mathbf{M}$, denoted by $\mathrm{ord}\left(\mathbf{M}\right)$, is the the least positive integer ℓ such that $\mathbf{M}^\ell=\mathbf{I}$, where $\mathbf{I}$ is the $n\times n$ identity matrix. It is easy to prove that $\mathrm{ord}\left(\mathbf{M}\right)$ is well-defined if and only if $\mathbf{M}\in GL_n\left(q\right)$. Moreover, I. Niven [11] proved that the multiplicative order of a matrix $\mathbf{M}\in GL_n\left(q\right)$ is less than or equal to q^n-1.

Given a finite field $\mathbb{F}_q$, we denote $\mathbb{F}_q\left[x\right]$ as the polynomial ring $\{\sum_{i=0}^n\alpha_i x^i : n\geq 0 \text{ and } \alpha_i\in\mathbb{F}_q \text{ for } 0\leq i\leq n\}$. The degree of a non zero $f\left(x\right)\in\mathbb{F}_q\left[x\right]$, denoted by $\deg\left(f\right)$, is defined as the exponent of highest power of x that appears in $f\left(x\right)$. A polynomial $f\left(x\right)$ is irreducible iff $f\left(x\right)$ does not have a trivial factor (a constant polynomial as a factor); otherwise, it is reducible. A polynomial $f\left(x\right)\in\mathbb{F}_q\left[x\right]$ with $\deg\left(f\right)=d\geq 1$ is called primitive polynomial over $\mathbb{F}_q$ if f is the polynomial over $\mathbb{F}_q$ of the least degree such that $f\left(\alpha\right)=0$ where α is a primitive element of $\mathbb{F}_{q^d}$. Clearly every primitive polynomial is also irreducible. For more comprehensive theoretical explanation, we refer the reader to [9], [10], [12].

III. DIFFIE-HELLMAN, ELLIPTIC CURVES DIFFIE-HELLMAN, AND MEGRELISHVILI PROTOCOL

A. *Diffie-Hellman and Elliptic Curves Diffie-Hellman Protocol*

Most modern cryptosystems are based on mathematical theories. These theories are including, but not always limited to, number theory, abstract algebra, and linear algebra. Diffie-Hellman key exchange is a simple procedure that allows two parties to exchange confidential information via an open channel. Originally, this protocol makes use of finite multiplicative group $\mathbb{F}_p^*$ [1], [10]. However, theoretically, the group chosen for the protocol can be any group $(G, *)$ written multiplicatively. For example, the elliptic curves Diffie-Hellman protocol operates on the elliptic curve over finite field $\mathbb{F}_q$, i.e. $E(\mathbb{F}_q)$, where q is a prime power. The group operation used in $E(\mathbb{F}_q)$ is the elliptic curve point addition in $E(\mathbb{F}_q)$. We refer the reader to [10] for more extensive theoretical explanation.

In general, Diffie-Hellman key agreement works as follows. Suppose two parties, namely Alice and Bob, want to exchange a key via an open channel. Initially, both of them need to agree on a finite group $(G, *)$ which is written multiplicatively. They choose an element $g \in G$ whose order is maximal in G. In practice, the value of g is chosen to be a primitive generator of G whenever it is possible. The next step is for Alice and Bob to choose respectively secret integers α and β, and subsequently compute $A = g^\alpha$ and $B = g^\beta$ as their corresponding public values. During the public exchange of values, Alice sends A to Bob whilst Bob sends B to Alice. Finally, Alice computes $A' = B^\alpha$ and similarly Bob computes $B' = A^\beta$. The value A' and B' are the shared secret keys. The correctness of Diffie-Hellman protocol is based on the fact that $A' = B^\alpha = g^{\beta\alpha} = g^{\alpha\beta} = A^\beta = B'$. The generic step-by-step procedure of this protocol is illustrated in Table I.

Public parameters creation	
A trusted party selects and publishes: a finite group $(G, *)$ and an element $g \in G$ whose order is maximal in G.	
Private keys generation	
Alice	**Bob**
Choose an integer α.	Choose an integer β.
Public exchange of values	
Compute $A = g^\alpha$. Alice sends A to Bob.	Compute $B = g^\beta$. Bob sends B to Alice.
Shared secret key computation	
Alice	**Bob**
Compute $A' = B^a$.	Compute $B' = A^\beta$.
The shared secret key is $A' = B'$.	

TABLE I: The generic step-by-step illustration of Diffie-Hellman key agreement.

During private keys generations, Alice and Bob respectively compute $A = g^\alpha$ and $B = g^\beta$. In the original version of Diffie-Hellman protocol, these values can be computed using fast modular exponentiation procedure in $\mathcal{O}(\log \alpha)$ and $\mathcal{O}(\log \beta)$ steps, respectively, where each step is a scalar operation in the associated finite field [10]. Using appropriate modification of fast exponentiation algorithm, it is not difficult to prove that for any integer x, the cost for calculating g^x in G is bounded by $\mathcal{O}(T_m \cdot \log \alpha)$, where T_m denotes number of steps required to perform one multiplication by g in G.

The security of Diffie-Hellman protocol or its variant on elliptic curves relates to the discrete logarithm problem (DLP) on the group $(G, *)$, that is, the problem of finding an integer x satisfying $g^x = h$ for $g, h \in G$. This problem can be solved using exhaustive search algorithm in $\mathcal{O}(|G|)$ steps, where each step consists of a multiplication by g. For example, the DLP for the conventional Diffie-Hellman protocol defined over $\mathbb{F}_q^*$ can be solved in $\mathcal{O}(q)$ steps, where each step is scalar multiplication in $\mathbb{F}_q$. Moreover, since the number of point in the elliptic curve $E(\mathbb{F}_q)$ for prime q satisfies $E(\mathbb{F}_q) = q + 1 - t_q$ where $|t_q| \leq 2\sqrt{q}$, the discrete logarithm problem in $E(\mathbb{F}_q)$ can also be solved in linear steps with respect to the size of the finite field used [10]. These two facts suggest that the private keys in both DHP or ECDHP can be obtained using exhaustive search procedure in $\mathcal{O}(q)$ steps, where q is the size of the finite field used in the protocols. However, faster algorithms do exists, such as Shanks' algorithm and

Pollard's ρ algorithm [10].

B. Megrelishvili Protocol

Megrelishvili key exchange protocol was first proposed by R. Megrelishvili, M. Chelidze, and K. Chelidze in [7]. This protocol is based on an adapted Diffie-Hellman protocol [1] which manipulates matrix exponentiation and vector-matrix multiplication over finite field. Based on [7], [13]–[15], we distinguish this protocol into two types. The first type solely uses matrix exponentiation and vector-matrix multiplication in its private keys computations, public exchange of values, and shared secret key computations. The second one incorporates matrix addition to the linear algebraic operations used in the first type. In this paper, we focus our research in the first type protocol.

In the following, we describe Megrelishvili key agreement according to [15]. Suppose two parties, namely Alice and Bob, want to exchange a key via an open channel. At the beginning, they have to agree on a finite field $\mathbb{F}_q$, an $n \times n$ nonsingular matrix $\mathbf{M}$ over $\mathbb{F}_q$, an n-dimensional non zero vector $\vec{v} \in \mathbb{F}_q^n$. The value of q, n, $\mathbf{M}$, and $\vec{v}$ are public knowledge. For the private keys computations, Alice and Bob correspondingly choose random (preferably positive) integers α and β and keep them secret, and separately generate $\mathbf{A} = \mathbf{M}^\alpha$ and $\mathbf{B} = \mathbf{M}^\beta$ as their personal private matrices. The matrix $\mathbf{M}$ is selected in such a way that $\operatorname{ord}(\mathbf{M})$ is sufficiently large. During the public exchange of values, Alice computes $\vec{a} = \vec{v}\mathbf{A}$ and sends $\vec{a}$ to Bob, while Bob computes $\vec{b} = \vec{v}\mathbf{B}$ and sends $\vec{b}$ to Alice. Afterward, Alice and Bob respectively calculate $\vec{a}' = \vec{b}\mathbf{A}$ and $\vec{b}' = \vec{a}\mathbf{B}$, the vector $\vec{a}'$ and $\vec{b}'$ are the shared secret keys. Observe that $\mathbf{AB} = \mathbf{M}^{\alpha+\beta} = \mathbf{M}^{\beta+\alpha} = \mathbf{BA}$, therefore $\vec{a}' = \vec{b}\mathbf{A} = \vec{v}\mathbf{BA} = \vec{v}\mathbf{AB} = \vec{a}\mathbf{B} = \vec{b}'$. The protocol is summarized in Table II.

Public parameters creation	
A trusted party selects and publishes: a finite field $\mathbb{F}_q$, a (large) integer n, an $n \times n$ nonsingular matrix $\mathbf{M}$ over $\mathbb{F}_q$, and an n-dimensional non zero vector $\vec{v}$ in $\mathbb{F}_q^n$.	
Private keys generation	
Alice	**Bob**
Choose an integer α. Calculate $\mathbf{A} = \mathbf{M}^\alpha$.	Choose an integer β. Calculate $\mathbf{B} = \mathbf{M}^\beta$.
Public exchange of values	
Compute $\vec{a} = \vec{v}\mathbf{A}$. Alice sends $\vec{a}$ to Bob.	Compute $\vec{b} = \vec{v}\mathbf{B}$. Bob sends $\vec{b}$ to Alice.
Shared secret key computation	
Alice	**Bob**
Compute $\vec{a}' = \vec{b}\mathbf{A}$.	Compute $\vec{b}' = \vec{a}\mathbf{B}$.
The shared secret key is $\vec{a}' = \vec{b}'$.	

TABLE II: Megrelishvili key exchange protocol according to [15].

We present a small numerical instance of Megrelishvili protocol in Example 1.

Example 1 Suppose Alice and Bob agree to use the finite field $\mathbb{F}_2$, a primitive nonsingular matrix $\mathbf{M} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$, and a vector $\vec{v} = (1, 1, 0)$. Alice chooses the secret integer $\alpha = 3$ and generate $\mathbf{A} = \mathbf{M}^\alpha = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ as her private matrix. At the same time Bob chooses the secret integer $\beta = 5$ and construct $\mathbf{B} = \mathbf{M}^\beta = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ as his private matrix. Alice sends Bob the vector $\vec{a} = \vec{v}\mathbf{A} = (1, 1, 1)$ and Bob sends Alice the vector $\vec{b} = \vec{v}\mathbf{B} = (1, 0, 0)$. This exchange is performed

over an insecure channel. Then Alice computes $\vec{a}' = \vec{b}\mathbf{A} = (0, 0, 1)$, while simultaneously Bob computes $\vec{b}' = \vec{a}\mathbf{B} = (0, 0, 1)$, the vector $\vec{a}' = \vec{b}'$ is their shared secret key.

IV. ELEMENTARY ALGORITHMS ANALYSIS

In this section, we measure the computational time complexities of several elementary algorithms related to Megrelishvili protocol. From now on, the term complexity refers to the computational time complexity, unless otherwise specified. We present the asymptotic complexity of each algorithm in $\mathcal{O}$ or Ω notation (see [8] for more detailed explanation).

A. Algorithm Analysis of The Private Keys Computations

In order to create the private matrices $\mathbf{A}$ and $\mathbf{B}$, Alice and Bob need to compute $\mathbf{M}^{\alpha}$ and $\mathbf{M}^{\beta}$, respectively. These values can be computed using two different elementary algorithms. The first algorithm is the standard (naive) exponentiation method and the second one is the fast exponentiation algorithm which utilizes the binary representation of the corresponding exponent (i.e. α or β). We describe the pseudocode of the standard exponentiation algorithm for calculating $\mathbf{M}^{\alpha}$ for an integer $\alpha \geq 1$ in Algorithm 1.

Algorithm 1 Standard Matrix Exponentiation Algorithm

Require: $\mathbf{M} \in M_n(q)$ and $\alpha \geq 1$
1: $\mathbf{N} \leftarrow \mathbf{M}$, $k \leftarrow 1$
2: **while** $k < \alpha$ **do**
3: $\quad \mathbf{N} \leftarrow \mathbf{NM}$ // $\mathbf{N} = \mathbf{M}^k$
4: $\quad k \leftarrow k + 1$ // increment k by 1
5: **end while**
Ensure: $k = \alpha$ and $\mathbf{N} = \mathbf{M}^{\alpha}$

The complexity of Algorithm 1 is analyzed in Theorem 1.

Theorem 1 *Suppose $\mathbf{M} \in M_n(q)$ and α is a positive integer. The complexity of Algorithm 1 is $\mathcal{O}(f(n) \cdot \alpha)$, where $f(n)$ stands for the complexity of the chosen matrix multiplication algorithm, and each step is a scalar operation in $\mathbb{F}_q$.*

Proof: Observe that the while loop of lines 2 - 5 of Algorithm 1 takes precisely α steps, where each step consists of a multiplication of two square matrices. Suppose $f(n)$ is the complexity of the chosen matrix multiplication algorithm (e.g. $f(n) = \mathcal{O}(n^3)$ for the conventional square matrix multiplication algorithm). If we assume that one scalar operation in $\mathbb{F}_q$ takes $\mathcal{O}(1)$ time, then the total computational cost for the entire algorithm is $f(n) \cdot \alpha = \mathcal{O}(f(n) \cdot \alpha)$. Thus, the complexity of Algorithm 1 is $\mathcal{O}(f(n) \cdot \alpha)$. □

For any $\mathbf{M} \in M_n(q)$ and any integer $\alpha \geq 1$, Algorithm 1 clearly outputs the correct $\mathbf{M}^{\alpha}$, yet this algorithm is highly impractical. In order to compute $\mathbf{M}^{\alpha}$ efficiently, we can modify the fast modular exponentiation algorithm for integer described in [10]. The idea is to use the binary representation of α to convert the calculation of $\mathbf{M}^{\alpha}$ into a succession of squarings and multiplications of matrices. Suppose the binary representation of α is a k-bits $\alpha_0\alpha_1 \ldots \alpha_{k-1}$, that is

$$\alpha = \alpha_0 2^0 + \alpha_1 2^1 + \cdots + \alpha_{k-1} 2^{k-1} = \sum_{i=0}^{k-1} \alpha_i 2^i, \tag{1}$$

where $\alpha_i \in \{0, 1\}$ for all $1 \leq i \leq n$. The value of $\mathbf{M}^{\alpha}$ can be computed in the following way

$$\mathbf{M}^{\alpha} = \mathbf{M}^{\sum_{i=0}^{k-1} \alpha_i 2^i} = \prod_{i=0}^{k-1} \mathbf{M}^{\alpha_i 2^i}. \tag{2}$$

Observe that $\mathbf{M}^{2^{i+1}} = \mathbf{M}^{2^i \cdot 2} = \left(\mathbf{M}^{2^i}\right)^2$, thus the sequence $\mathbf{M}^{2^0}, \mathbf{M}^{2^1}, \ldots, \mathbf{M}^{2^{n-1}}$ can be computed efficiently using successive squaring. Moreover, the calculation of $\mathbf{M}^{\alpha}$ can be performed using k successive squarings and k multiplications of matrices. The pseudocode for the fast matrix exponentiation algorithm is explained in Algorithm 2. The complexity of Algorithm 2 is analyzed in Theorem 2.

Algorithm 2 Fast Matrix Exponentiation Algorithm

Require: $\mathbf{M} \in M_n(q)$ and $\alpha \geq 1$
1: $\mathbf{N} \leftarrow \mathbf{I}$
2: **while** $\alpha > 0$ **do**
3: **if** $\alpha \bmod 2 = 1$ **then**
4: $\mathbf{N} \leftarrow \mathbf{NM}$
5: **end if**
6: $\mathbf{M} \leftarrow \mathbf{M}^2$
7: $\alpha \leftarrow \alpha \operatorname{div} 2$
8: **end while**
Ensure: $\alpha = 0$ and $\mathbf{N} = \mathbf{M}^{\alpha}$

Theorem 2 *Suppose $\mathbf{M} \in M_n(q)$ and α is a positive integer. The complexity of Algorithm 2 is $\mathcal{O}(f(n) \cdot \log \alpha)$, where $f(n)$ stands for the complexity of the chosen matrix multiplication algorithm, and and each step is a scalar operation in $\mathbb{F}_q$.*

Proof: The while loop of lines 2 - 8 of Algorithm 2 is executed whenever the value of α is positive. In each iteration, the value of α is reduced to half. Suppose the binary representation of α is a k-bits $\alpha_0\alpha_1 \ldots \alpha_{k-1}$, then $\alpha = \sum_{i=0}^{k-1} \alpha_i 2^i$, and consequently the number of while loop iterations of lines 2 - 8 is at most k. Since $\alpha = \mathcal{O}\left(2^k\right)$, then the total number of while loop iterations in Algorithm 2 is at most $\mathcal{O}(\log_2(\alpha))$. Assuming that one scalar operation in $\mathbb{F}_q$ requires $\mathcal{O}(1)$ time, the complexity of the entire algorithm is $f(n) \cdot \mathcal{O}(\log_2 \alpha) = \mathcal{O}(f(n) \cdot \log \alpha)$. □

The complexities of Algorithm 1 and Algorithm 2 depend on the chosen square matrix multiplication algorithm. The complexity of the standard square matrix multiplication algorithm is $\mathcal{O}\left(n^3\right)$ [8]. Thus, if this algorithm is implemented in Algorithm 1 and Algorithm 2, then the complexity of each algorithm becomes $\mathcal{O}\left(n^3 \cdot \alpha\right)$ and $\mathcal{O}\left(n^3 \cdot \log \alpha\right)$, respectively. Obviously, Algorithm 2 is more preferable to Algorithm 1. The asymptotic running time of Algorithm 2 can be optimized using faster matrix multiplication algorithm. Currently, the asymptotically fastest known deterministic algorithm for square matrix multiplication is due to A. Ambainis, Y. Filmus, and F. Gall [16]. The running time of this algorithm is $\mathcal{O}\left(n^{2.3728639}\right)$, yet the constant coefficient hidden by the $\mathcal{O}$ notation is enormous that the implementation of this algorithm and some of its predecessors (such as Coppersmith-Winograd's Algorithm [17], Stothers' Algorithm [18], and Williams' Algorithm [19]) are considered impractical. Another deterministic algorithm which is asymptotically faster than the conventional method is the well-known Strassen's Algorithm [20] which runs in $\mathcal{O}\left(n^{\lg 7}\right)$ time. However, Strassen's Algorithm is not often the best choice for square matrix multiplication due to numerical stability and space complexity issue [8]. If the utilization of non deterministic algorithm for square matrix multiplication is allowed, the complexity of Algorithm 2 can be reduced to $\mathcal{O}\left(n^2 \cdot \log \alpha\right)$ by Freivalds' Algorithm [21].

To exchange the public values (the vector $\vec{a}$ and $\vec{b}$), each party needs to compute $\vec{a} = \vec{v}\mathbf{A}$ and $\vec{b} = \vec{v}\mathbf{B}$, respectively. Each of these operation can be considered as a matrices multiplication of size $1 \times n$ and $n \times n$ whenever $\vec{v} \in \mathbb{F}_q^n$ and $\mathbf{A}, \mathbf{B} \in M_n(q)$. These operations can be carried out using conventional matrix multiplication algorithm in $\mathcal{O}(1 \cdot n \cdot n) = \mathcal{O}\left(n^2\right)$ steps, where each step is a scalar operation in $\mathbb{F}_q$. As a result, with the assumption that the cost for square matrix multiplication satisfies $f(n) = \Omega\left(n^2\right)$ [8], the total cost for constructing a private key and its corresponding vector to be exchanged is $\mathcal{O}(f(n) \cdot \log \alpha) + \mathcal{O}\left(n^2\right) = \mathcal{O}(f(n) \cdot \log \alpha)$.

To conclude the key exchange agreement, each party needs to compute their shared secret key. Each of Alice and Bob needs to calculate $\vec{a}' = \vec{b}\mathbf{A}$ and $\vec{b}' = \vec{a}\mathbf{B}$, respectively. If $\vec{a}, \vec{b} \in \mathbb{F}_q^n$ and $\mathbf{A}, \mathbf{B} \in M_n(q)$, then each of the calculation is a matrices multiplication of size $1 \times n$ and $n \times n$. Therefore, the shared

secret key can be computed in $\mathcal{O}\left(1 \cdot n \cdot n\right) = \mathcal{O}\left(n^2\right)$ steps, where each step is a scalar operation in $\mathbb{F}_q$. The summary for algorithms complexities in private keys generations, public exchange of values, and shared secret key computations is described in Table III.

Algorithms	Mathematical operations	Asymptotic complexities
Private key generation	$\mathbf{M}^{\alpha}$ ($\mathbf{M} \in M_n\left(q\right)$ and $\alpha \in \mathbb{N}$)	Upper bound: $\mathcal{O}\left(f\left(n\right) \cdot \log \alpha\right)$ Lower bound: $\Omega\left(n^2 \log \alpha\right)$
Public exchange of value	$\vec{v}\mathbf{A}$ where $\vec{v} \in \mathbb{F}_q^n$ and $\mathbf{A} = \mathbf{M}^{\alpha}$	Upper bound: $\mathcal{O}\left(n^2\right)$ Lower bound: $\Omega\left(n^2\right)$
Shared secret key computation	$\vec{b}\mathbf{A}$ where $\vec{b} \in \mathbb{F}_q^n$ and $\mathbf{A} = \mathbf{M}^{\alpha}$	Upper bound: $\mathcal{O}\left(n^2\right)$ Lower bound: $\Omega\left(n^2\right)$

TABLE III: Elementary algorithms complexities in Megrelishvili key exchange protocol. The function $f(n)$ denotes the asymptotic complexity for the chosen matrix multiplication algorithm. The mathematical operations involved are observed from Alice's perspective.

B. Algorithm Analysis of Public Matrices Construction

The explicit construction method of the public matrices for Megrelishvili protocol was first described in [13]. This method is also reviewed in [15], [22]. The technique of generating the public matrices in [13], [15], [22] is easy, yet it only deals with the case where the vector space is defined over $\mathbb{F}_2$ and the dimension of the vector space is odd. The original construction method is explained as follows. Suppose $\mathbf{M}_n$ denotes the public matrices of size n, the matrix $\mathbf{M}_n$ for odd values of n is composed using following recurrences

$$\mathbf{M}_3 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{M}_5 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & & & & 0 \\ 0 & & \mathbf{M}_3 & & 1 \\ 1 & & & & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{M}_7 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & & & & & & 0 \\ 0 & & & & & & 1 \\ 1 & & & \mathbf{M}_5 & & & 0 \\ 0 & & & & & & 1 \\ 1 & & & & & & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix},$$

$$\cdots$$

$$\mathbf{M}_{n+2} = \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots & 0 & 1 \\ 1 & & & & & & 0 \\ 0 & & & & & & 1 \\ 1 & & & \mathbf{M}_n & & & 0 \\ \vdots & & & & & & \vdots \\ & & & & & & 0 \\ 0 & 1 & 0 & 1 & \cdots & 1 & 0 \end{bmatrix}. \tag{3}$$

It is clear that the construction of $\mathbf{M}_n$ can be performed using a doubly-nested for loops algorithm, hence the creation of $\mathbf{M}_n$ requires $\mathcal{O}\left(n^2\right)$ steps.

Another technique to construct the public matrices is by utilizing the companion matrices of primitive polynomials over finite fields [23], [24]. This method is not limited to a particular finite field and the size of the matrices produced are arbitrary. Suppose $f\left(x\right) = x^n + \sum_{i=0}^{n-1} a_i x_i$ is a monic polynomial of

degree n over $\mathbb{F}_q[x]$. The companion matrix $\mathbf{C}$ for $f(x)$ is defined as $\mathbf{C} = [\mathbf{C}_{ij}]$, where

$$\mathbf{C}_{ij} = \begin{cases} -a_{i-1}, & \text{if } 1 \leq i \leq n \text{ and } j = n \\ 1, & \text{if } i - j = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The matrix $\mathbf{C}$ of size $n \times n$ can be created using a doubly-nested for loops algorithm which runs in $\mathcal{O}(n^2)$ steps. Furthermore, this matrix can be used as a public matrix for Megrelishvili protocol.

However, using a single particular public matrix in a key exchange protocol for a fixed value of vector space dimension is insecure. Therefore, the matrices that are produced using recurrence equation (3) and companion matrix definition in equation (4) can not be used straightforwardly. We present a randomized algorithm to construct public matrix of size n in Algorithm 3.

Algorithm 3 Randomized Construction of Public Matrix

Require: a matrix $\mathbf{N}$ of size n that is generated using equation (3) or (4).
1: randomly generates $\mathbf{Q} \in GL_n(q)$
2: $\mathbf{M} \leftarrow \mathbf{QNQ}^{-1}$ // $\mathbf{M} = \mathbf{QNQ}^{-1}$
Ensure: $\mathbf{M}$ is a matrix which satisfies $\operatorname{ord}(\mathbf{M}) = \operatorname{ord}(\mathbf{N})$

Algorithm 3 requires a matrix that is generated using equation (3) or (4). Suppose $\mathbf{N}$ is an input of Algorithm 3, the output of this algorithm is a matrix $\mathbf{M}$ which satisfies $\operatorname{ord}(\mathbf{M}) = \operatorname{ord}(\mathbf{N})$. The correctness of this algorithm is explained in Theorem 3.

Theorem 3 *Let* $\mathbf{M}$ *and* $\mathbf{N}$ *be two matrices satisfying* $\mathbf{M} = \mathbf{QNQ}^{-1}$, *then* $\operatorname{ord}(\mathbf{M}) = \operatorname{ord}(\mathbf{N})$.

Proof: Suppose $\operatorname{ord}(\mathbf{N}) = \ell$, then ℓ is the least positive integer satisfying $\mathbf{N}^\ell = \mathbf{I}$. By mathematical induction on ℓ, we have $\mathbf{M}^\ell = \left(\mathbf{QNQ}^{-1}\right)^\ell = \mathbf{QN}^\ell\mathbf{Q}^{-1} = \mathbf{QIQ}^{-1} = \mathbf{I}$, and therefore $\operatorname{ord}(\mathbf{M}) \leq \ell$. Suppose $\operatorname{ord}(\mathbf{M}) = k < \ell$, then, by mathematical induction on k, we have $\mathbf{M}^k = \left(\mathbf{QNQ}^{-1}\right)^k = \mathbf{QN}^k\mathbf{Q}^{-1}$, and consequently $\mathbf{N}^k = \mathbf{Q}^{-1}\mathbf{M}^k\mathbf{Q}$. The supposition that $\operatorname{ord}(\mathbf{M}) = k$ gives $\mathbf{M}^k = \mathbf{I}$, and thus $\mathbf{N}^k = \mathbf{Q}^{-1}\mathbf{M}^k\mathbf{Q} = \mathbf{I}$, which means $\operatorname{ord}(\mathbf{N}) \leq k < \ell$. This contradicts our previous assumption that $\operatorname{ord}(\mathbf{N}) = \ell$. Therefore, it must be the case that $\operatorname{ord}(\mathbf{M}) = \operatorname{ord}(\mathbf{N}) = \ell$. □

Observe that Algorithm 3 only performs one mathematical calculation of line 2, i.e. the computation of $\mathbf{QNQ}^{-1}$. This computation consists of two multiplications of matrices and one matrix inversion. Since matrix inversion is no harder than matrix multiplication [8, Theorem 28.2], the total computational cost of line 2 is $\mathcal{O}(2 \cdot f(n) + f(n)) = \mathcal{O}(f(n))$, where $f(n)$ denotes the complexity for the chosen matrix multiplication algorithm. The running time of Algorithm 3 also depends on the complexity of generating a random nonsingular matrix $\mathbf{Q}$ of line 1. If this line can be executed in $\mathcal{O}(g(n))$ time, then the running time of the algorithm becomes $\mathcal{O}(f(n) + g(n))$.

We propose an elementary method to construct random nonsingular matrices over $\mathbb{F}_q$ in Algorithm 4. This algorithm requires a positive integer n as an input and it produces a random nonsingular $n \times n$ matrix $\mathbf{Q}$.

The correctness and complexity analysis of Algorithm 4 is explained in Theorem 4.

Theorem 4 *Given a positive integer* n, *Algorithm 4 yields a random nonsingular matrix* $\mathbf{Q}$ *in* $\mathcal{O}(n^2)$ *steps.*

Proof: The for loop of lines 1 - 11 is a construction of an upper triangular matrix $\mathbf{Q}$ which satisfies following definition

$$\mathbf{Q} = \mathbf{Q}[i,j] \text{ where } \mathbf{Q}[i,j] = \begin{cases} \text{a random non zero } a \in \mathbb{F}_q, & \text{if } i = j \\ \text{a random } a \in \mathbb{F}_q, & \text{if } i < j \\ 0, & \text{if } i > j. \end{cases} \tag{5}$$

Therefore, after line 11 is executed, Algorithm 4 yields an upper triangular matrix $\mathbf{Q}$ whose diagonal elements are non zero. Consequently, $\det(\mathbf{Q}) = \prod_{i=1}^{n} \mathbf{Q}[i,i] \neq 0$, and hence the matrix $\mathbf{Q}$ is nonsingular. Each of line 12 and 13 respectively performs a consecutive permutation of the rows and the columns of

Algorithm 4 Random Nonsingular Matrix Generation

Require: a positive integer n.

1: **for** $i \leftarrow 1$ to n **do**
2: **for** $j \leftarrow 1$ to n **do**
3: **if** $i = j$ **then**
4: $\mathbf{Q}[i,j] \leftarrow \mathrm{Random.Element}(\mathbb{F}_q^*)$ // $\mathbf{Q}[i,j]$ is a non zero element if $i = j$
5: **else if** $i < j$ **then**
6: $\mathbf{Q}[i,j] \leftarrow \mathrm{Random.Element}(\mathbb{F}_q)$ // $\mathbf{Q}[i,j]$ is an arbitrary element if $i < j$
7: **else**
8: $\mathbf{Q}[i,j] \leftarrow 0$ // $\mathbf{Q}[i,j]$ is zero if $i > j$
9: **end if**
10: **end for**
11: **end for**
12: $\mathbf{Q} \leftarrow \mathrm{Permute.row}(\mathbf{Q})$ // permutes the rows of Q
13: $\mathbf{Q} \leftarrow \mathrm{Permute.col}(\mathbf{Q})$ // permutes the columns of Q

Ensure: $\mathbf{Q}$ is a nonsingular matrix of size n.

the matrix. Since the singularity of a matrix is invariant under rows and columns permutations [12], the matrix $\mathbf{Q}$ produced after line 13 is nonsingular.

Observe that the doubly nested loop of lines 1 - 11 yields an $\Omega\left(n^2\right)$ lower bound for on the running time of Algorithm 4 . Assuming that each of $\mathrm{Random.Element}(\mathbb{F}_q^*)$ and $\mathrm{Random.Element}(\mathbb{F}_q)$ subroutine takes $\mathcal{O}(1)$ time and each of $\mathrm{Permute.row}(\mathbf{Q})$ and $\mathrm{Permute.col}(\mathbf{Q})$ subroutine takes $\mathcal{O}(n)$ time (see [8] for more detailed explanation concerning randomized algorithms), the overall complexity of this algorithm is $\mathcal{O}\left(n^2 + 2 \cdot n + 2 \cdot 1\right) = \mathcal{O}\left(n^2\right)$ steps, where each step is a scalar operation in $\mathbb{F}_q$. □

Since the creation of a random nonsingular matrix $\mathbf{Q}$ can be performed in $\mathcal{O}\left(n^2\right)$ steps, the running time of Algorithm 3 becomes $\mathcal{O}\left(f(n) + n^2\right)$. Assuming that the complexity for square matrix multiplication satisfies $f(n) = \Omega\left(n^2\right)$ [8], the running time of Algorithm 3 turns into $\mathcal{O}(f(n))$. This means the complexity of this algorithm depends solely on the matrix multiplication procedure of line 2.

C. Algorithm Analysis of Exhaustive Search Attack

In this section, we analyze the security of Megrelishvili key-agreement protocol against an elementary exhaustive search attack. We introduce the concept of Megrelishvili matrix-vector problem as the linear algebraic problem that underlies the security of this protocol in Definition 1.

Definition 1 Let $\mathbb{F}_q^n$ be a vector space over the finite field $\mathbb{F}_q$, $\vec{v}$ and $\vec{w}$ are two vectors in $\mathbb{F}_q^n$, and $\mathbf{M}$ is an $n \times n$ nonsingular matrix over $\mathbb{F}_q$. The Megrelishvili vector-matrix problem (MVMP for short) is the problem of determining an integer α (if such integer exists) which satisfies $\vec{v}\mathbf{M}^\alpha = \vec{w}$.

Megrelishvili matrix-vector problem is similar to the well-known discrete logarithm problem. One palpable difference is that the exponentiation in this problem is applied to a square matrix and the quantities that are compared on each side of the equal sign are vectors. These vectors are not the element of a multiplicative group. We present an exhaustive search procedure which solves this problem in Algorithm 5. In this algorithm we use the fact that the multiplicative order of a matrix $\mathbf{M} \in GL_n(q)$ is no more than $q^n - 1$ [11].

It is apparent that, if there is an integer α in $1 \leq \alpha \leq q^n - 1$ that satisfies $\vec{v}\mathbf{M}^\alpha = \vec{w}$, then Algorithm 5 outputs α. Therefore Algorithm 5 gives a trivial bound for MVMP.

Theorem 5 *Let $\mathbb{F}_q^n$ be a vector space over the finite field $\mathbb{F}_q$, $\vec{v}$ and $\vec{w}$ are two vectors in $\mathbb{F}_q^n$, and $\mathbf{M}$ is an $n \times n$ nonsingular matrix over $\mathbb{F}_q$. Then the Megrelishvili vector-matrix problem $\vec{v}\mathbf{M}^\alpha = \vec{w}$ can be solved in $\mathcal{O}(f(n) \cdot q^n)$ steps, where $f(n)$ stands for the complexity of the chosen matrix multiplication algorithm and each step consists of scalar operations in $\mathbb{F}_q$.*

Algorithm 5 Exhaustive Algorithm for Solving MVMP

Require: $\mathbf{M} \in M_n(q)$ and $\vec{v}, \vec{w} \in \mathbb{F}_q^n$
1: $\mathbf{Q} \leftarrow \mathbf{M}$, $\alpha \leftarrow 1$
2: **while** $\alpha \leq q^n - 1$ and $\vec{v}\mathbf{Q} \neq \vec{w}$ **do**
3: $\alpha \leftarrow \alpha + 1$ // increment α by 1
4: $\mathbf{Q} \leftarrow \mathbf{QM}$ // $\mathbf{Q} = \mathbf{M}^\alpha$
5: **end while**
Ensure: $\alpha = q^n$ or $\vec{v}\mathbf{Q} = \vec{w}$

Proof: The line 4 of Algorithm 5 is a multiplication of two $n \times n$ matrices. Suppose $f(n)$ is the complexity of the chosen matrix multiplication algorithm. Since there are at most $q^n - 1$ matrix multiplications in Algorithm 5, the total number of scalar operations in the algorithm is $\mathcal{O}(f(n) \cdot q^n - 1) = \mathcal{O}(f(n) \cdot q^n)$. □

The number of iterations required to find α from the equation $\vec{v}\mathbf{M}^\alpha = \vec{w}$ depends on the value of $\vec{v}$, $\vec{w}$, and $\operatorname{ord}(\mathbf{M})$. Moreover, whenever the value of $\operatorname{ord}(\mathbf{M})$ is known, the expression $q^n - 1$ in Algorithm 5 can be altered to $\operatorname{ord}(\mathbf{M})$, and therefore the trivial bound for MVMP becomes $\mathcal{O}(f(n) \cdot \operatorname{ord}(\mathbf{M}))$. This means the number of iterations performed to check whether $\vec{v}\mathbf{M}^\alpha = \vec{w}$ can be reduced to $\operatorname{ord}(\mathbf{M})$. For example, let $\vec{v} = (0,0,1)$, $\vec{w} = (1,1,1)$, and $\mathbf{M} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$. The order of $\mathbf{M}$ is 3 and $\vec{v}\mathbf{M} = (1,0,0)$, $\vec{v}\mathbf{M}^2 = (1,1,0)$, $v\mathbf{M}^3 = (0,0,1)$, and thus the equation $\vec{v}\mathbf{M}^\alpha = \vec{w}$ has no solution.

V. COMPLEXITIES COMPARISON AMONG CONVENTIONAL DHP, ECDHP, AND MEGRELISHVILI PROTOCOL

In this section, we discuss the complexities comparison among conventional Diffie-Hellman protocol (DHP), elliptic curves Diffie-Hellman protocol (ECDHP), and Megrelishvili protocol. We divide each protocol into three main phases: private key generation, public exchange of value, and further private computation. The complexities of these phases in each protocol are compared to each other. In addition, we also contrast the number of scalar operations required to reveal the secret integer in each key exchange scheme using exhaustive search attack.

All protocols are considered to work over the finite field $\mathbb{F}_q$. To simplify our analysis, we restrict our observation to the case where q is prime. The dimension of the vector space used in Megrelishvili protocol is denoted by n. In our analysis all scalar operations in $\mathbb{F}_q$ are assumed to take $\mathcal{O}(1)$ steps.

A. Comparison for Private Key Generation

The private keys for the conventional DHP and ECDHP are simply integers (see Table I) which can be selected in $\mathcal{O}(1)$ steps. On the other hand, the private keys for Megrelishvili protocol consist of two values, a random integer α and a matrix $\mathbf{A}$ which is obtained from exponentiating the public matrix $\mathbf{M}$ with α. Although the selection of α can be performed in $\mathcal{O}(1)$ steps, the calculation of $\mathbf{A} = \mathbf{M}^\alpha$ is time consuming. The upper bound of this private matrix generation is $\mathcal{O}\left(n^3 \log \alpha\right)$ steps, whereas the lower bound is $\Omega\left(n^2 \log \alpha\right)$ steps (see Section IV-A). However, Alice may decide to generate $\mathbf{A}$ later during the public exchange of value. In this condition, Alice's private key is simply an integer α, which similar to the private key in DHP and ECDHP.

B. Comparison for Public Exchange of Value

In the conventional DHP, Alice needs to calculate g^α as her public value. This computation can be performed in $\mathcal{O}(\log \alpha)$ steps using fast modular exponentiation algorithm [10]. Analogously, in ECDHP, Alice needs to calculate αP where $P \in E(\mathbb{F}_q)$ as her public value. The computation of this value can be carried out using double-and-add algorithm which takes $\mathcal{O}(\log \alpha)$ steps [10]. Hence, in both DHP

and ECDHP, the public values to be exchanged can be computed in $\mathcal{O}(\log \alpha)$ steps. Since in DHP and ECDHP typically $\alpha \leq q$, the upper bounds for public value calculations in both DHP and ECDHP become $\mathcal{O}(\log q)$ steps [10].

In Megrelishvili protocol, Alice has to calculate $\vec{v}\mathbf{A}$ as her public value. Whenever the matrix $\mathbf{A}$ is already computed during private key generation, this computation can be performed in $\mathcal{O}\left(n^2\right)$ steps, where n is the dimension of the vector space used (see Section IV-A). However, Alice may choose not to calculate $\mathbf{A}$ during the private key generation. If this condition happens, then Alice needs to calculate $\vec{v}\mathbf{M}^{\alpha}$ as her public value. This calculation consists of a matrix exponentiation $\mathbf{M}^{\alpha}$ which requires $\mathcal{O}(f(n)\log\alpha)$ steps (see Theorem 1) and a vector-matrix multiplication which takes $\mathcal{O}\left(n^2\right)$ steps (see Section IV-A). The function $f(n)$ stands for the complexity of the chosen matrix multiplication algorithm. Therefore, under the assumption that $f(n) = \Omega\left(n^2\right)$ (see [8]), the total number of scalar computations to obtain $\vec{v}\mathbf{M}^{\alpha}$ becomes $\mathcal{O}(f(n)\cdot\log\alpha) + \mathcal{O}\left(n^2\right) = \mathcal{O}(f(n)\cdot\log\alpha)$. Since in Megrelishvili protocol typically $\alpha \leq q^n - 1$, the upper bound for this computation becomes $\mathcal{O}(f(n)\cdot\log(q^n-1)) = \mathcal{O}(n\cdot f(n)\log q)$. For example, if we implement the naive matrix multiplication algorithm to calculate $\vec{v}\mathbf{M}^{\alpha}$, the upper bound becomes $\mathcal{O}\left(n^4\log q\right)$.

C. Comparison for Further Private Computation

Alice determines the shared secret key in conventional DHP by exponentiating the public value from Bob. That is, she calculates B^{α} where B is Bob's public value. This computation can be performed by fast modular exponentiation in $\mathcal{O}(\log\alpha)$ steps. Similarly, in ECDHP, Alice determines the shared secret key by calculating αQ_B, where $Q_B \in E(\mathbb{F}_q)$ is Bob's public point (see [10, Table 6.5]). The computation of αQ_B can be carried out using double-and-add algorithm in $\mathcal{O}(\log\alpha)$ steps [10]. Since in DHP and ECDHP typically $\alpha \leq q$, the upper bounds for public value calculations in both DHP and ECDHP become $\mathcal{O}(\log q)$ steps [10].

In Megrelishvili protocol, Alice determines the shared secret key by computing $\vec{b}\mathbf{A}$, where $\vec{b} \in \mathbb{F}_q^n$ is Bob's public vector. Since $\mathbf{A}$ has previously been computed during private key generation or public exchange of value, the calculation of $\vec{b}\mathbf{A}$ requires $\mathcal{O}\left(n^2\right)$ steps, where n is the dimension of the vector space used. However, whenever the value q and n satisfy $n \geq \sqrt{\log q}$ (which happens in some cases, see [7]), the computational complexity for calculating $\vec{b}\mathbf{A}$ clearly higher than the computational complexity for calculating B^{α} and αQ_B in its DHP and ECDHP counterpart.

D. Comparison for Exhaustive Search Attack

It is obvious that the discrete logarithm problem (DLP) in $\mathbb{F}_q$ and $E(\mathbb{F}_q)$ can be solved using exhaustive search algorithm in $\mathcal{O}(q)$ steps, where each step is a scalar operation in $\mathbb{F}_q$ [10]. From Theorem 5, we see that the upper bound to obtain the secret integer in Megrelishvili protocol using similar exhaustive search techniques requires $\mathcal{O}(f(n)\cdot q^n)$ steps. This means the complexity of solving Megrelishvili matrix-vector problems is higher by a factor of $f(n)\,q^{n-1}$ than its DLP and ECDLP counterpart. Intuitively, this condition means Megrelishvili protocol is "more secure" than DHP and ECDHP against exhaustive search attack.

We summarize our analysis in this section in Table IV.

VI. CONCLUSION AND FUTURE WORKS

In this article, we explore the complexities of several elementary algorithms involved in Megrelishvili protocol. We give asymptotic bounds for the running time of elementary algorithms utilized during the private keys generations. These asymptotic bounds are summarized in Table III. We also propose an elementary randomized procedure for generating the public matrices for Megrelishvili protocol in Algorithm 3 and show that the complexity of this algorithm depends exclusively on the matrix multiplication method it chooses to implement. Finally, we introduce the notion of Megrelishvili vector-matrix problem (MVMP) as an underlying mathematical problem for the security of Megrelishvili protocol. We gives an $\mathcal{O}(f(n)\cdot\operatorname{ord}(\mathbf{M}))$ trivial bound for MVMP, where $\operatorname{ord}(\mathbf{M})$ is the order of an $n\times n$ public matrix

Computation	Conventional DHP over $\mathbb{F}_q$		ECDHP over $E(\mathbb{F}_q)$		Megrelishvili protocol over $\mathbb{F}_q^n$	
	Formula	Complexity	Formula	Complexity	Formula	Complexity
Private key generation	Selecting an integer α.	$\mathcal{O}(1)$	Selecting an integer α.	$\mathcal{O}(1)$	Selecting an integer α. Calculate $\mathbf{A} = \mathbf{M}^{\alpha}$.	$\mathcal{O}(1)$ Upper bound: $\mathcal{O}\left(n^3 \log \alpha\right)$. Lower bound: $\Omega\left(n^2 \log \alpha\right)$.
Public exchange of value	$A = g^{\alpha}$	$\mathcal{O}(\log \alpha)$	$Q_A = \alpha P$	$\mathcal{O}(\log \alpha)$	$\vec{a} = \vec{v}\mathbf{A}$ $\vec{a} = \vec{v}\mathbf{M}^{\alpha}$	$\mathcal{O}\left(n^2\right)$ $\mathcal{O}\left(f(n) \log \alpha\right)$
Further private computation	$A' = B^{\alpha}$	$\mathcal{O}(\log \alpha)$	$Q'_A = \alpha Q_B$	$\mathcal{O}(\log \alpha)$	$\vec{a}' = \vec{b}\mathbf{A}$	$\mathcal{O}\left(n^2\right)$
Exhaustive search attack	Solving $h = g^x$ for x.	$\mathcal{O}(q)$	Solving $Q = xP$ for x.	$\mathcal{O}(q)$	Solving $\vec{w} = \vec{v}\mathbf{M}^x$ for x.	Upper bound: $\mathcal{O}\left(f(n) \cdot q^{n-1}\right)$. Lower bound: $\Omega\left(n^2 \cdot \mathrm{ord}(\mathbf{M})\right)$.

TABLE IV: Summary of complexities comparison among DHP, ECDHP, and Megrelishvili protocol. The function $f(n)$ denotes the asymptotic complexity for the chosen matrix multiplication algorithm. The private key computation, public exchange of value, and further private computation are observed from Alice's perspective.

used in the protocol and $f(n)$ is the complexity for the chosen matrix multiplication algorithm.

The summary of complexities comparison among conventional Diffie-Hellman Protocol (DHP), elliptic curves Diffie-Hellman protocol (ECDHP), and Megrelishvili protocol is summarized in Table IV. Compared to DHP or its variant on elliptic curves (ECDHP), the complexities of the private keys generations in Megrelishvili protocol are higher. Moreover, the computational complexities of the public exchange of values and further private computations in this protocol are quadratic with respect to dimension of the vector space used. These two reasons are probably the reasons why the implementation of Megrelishvili protocol is still limited.

In Section V-D, we show that the complexity of solving MVMP by exhaustive search procedure is higher by a factor of $f(n)\, q^{n-1}$ than its discrete logarithm problem (DLP) and elliptic curves version (ECDLP) counterpart. This condition implies that the number of scalar calculations required to solve MVMP is greater than the number of scalar calculations required to solve DLP and ECDLP. Intuitively, this result also indicates that Megrelishvili protocol is "more secure" than DHP and ECDHP against exhaustive search attack. Nevertheless, this outcome also prompts a scientific question whether Megrelishvili protocol is theoretically and practically safer than the DHP or ECDHP. To our knowledge, any non-trivial algorithm for solving MVMP has not been proposed.

ACKNOWLEDGMENT

The author would like to thank both reviewers for their feedbacks.

REFERENCES

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644–654, 1976.
[2] A. Joux, "A one round protocol for tripartite Diffie-Hellman," in *Algorithmic number theory*. Springer, 2000, pp. 385–393.
[3] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An efficient protocol for authenticated key agreement," *Designs, Codes and Cryptography*, vol. 28, no. 2, pp. 119–134, 2003.
[4] A. J. Menezes and Y.-H. Wu, "The discrete logarithm problem in $GL(n, q)$," *Ars Combinatoria*, vol. 47, pp. 23–32, 1997.
[5] J. B. Nelson, "The Diffie-Hellman key exchange in matrices over a field and a ring," Master's thesis, Texas Tech University, 2003.
[6] J. N. Doliskani, E. Malekian, and A. Zakerolhosseini, "A cryptosystem based on the symmetric group S_n," *International Journal of Computer Science and Network Security*, vol. 8, no. 2, pp. 226–234, 2008.

[7] 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.
[8] T. H. Cormen, C. E. Leiserson, R. E. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.
[9] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*. Cambridge university press, 1994.
[10] J. Hoffstein, J. C. Pipher, and J. H. Silverman, *An introduction to mathematical cryptography*, 2nd ed. Springer, 2014.
[11] I. Niven, "Fermat's theorem for matrices," *Duke Mathematical Journal*, vol. 15, no. 3, pp. 823–826, 1948.
[12] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.
[13] 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.
[14] 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.
[15] A. Beletsky, A. Beletsky, and R. Kandyba, "Matrix analogues of the Diffie-Hellman protocol," in *ICTERI*, 2013, pp. 352–359.
[16] A. Ambainis, Y. Filmus, and F. L. Gall, "Fast matrix multiplication: limitations of the laser method," *arXiv preprint arXiv:1411.5414*, 2014.
[17] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM, 1987, pp. 1–6.
[18] A. J. Stothers, "On the complexity of matrix multiplication," Ph.D. dissertation, The University of Edinburgh, 2010.
[19] V. V. Williams, "Multiplying matrices in $\mathcal{O}(n^{2.373})$time," *preprint, http://theory. stanford. edu/˜ virgi/matrixmult-f. pdf, Stanford, CA*, 2014.
[20] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969.
[21] R. Freivalds, "Probabilistic machines can use less running time," in *IFIP congress*, vol. 839, 1977, p. 842.
[22] 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.
[23] A. J. Beletsky and A. A. Beletsky, "Synthesis of primitive matrices over a finite Galois fields and their applications (*in Russian*)," *Information Technology in Education: Collected Works*, vol. 13, pp. 23–43, 2012.
[24] ——, "Conveyor analog matrix for Diffie-Hellman protocol (*in Russian*)," *Information Technologies in Education*, vol. 14, pp. 11–16, 2013.