J. Algebra Comb. Discrete Appl.
4(3) • 247–260

Received: 23 January 2016
Accepted: 23 January 2017

Journal of Algebra Combinatorics Discrete Structures and Applications

Gaussian elimination in split unitary groups with an application to public-key cryptography*

Research Article

Ayan Mahalanobis, Anupam Singh

Abstract: Gaussian elimination is used in special linear groups to solve the word problem. In this paper, we extend Gaussian elimination to split unitary groups. These algorithms have an application in building a public-key cryptosystem, we demonstrate that.

2010 MSC: 20H30, 94A60

Keywords: Unitary groups, Gaussian elimination, Row-column operations

1. Introduction

Gaussian elimination is a very old theme in computational mathematics. It was developed to solve linear simultaneous equations. The modern day matrix theoretic approach was developed by John von Neumann and the popular textbook version by Alan Turing. Gaussian elimination has many applications and is a very well known mathematical method. We will not elaborate on it any further, but will refer an interested reader to a nice article by Grcar [10]. The way we look at Gaussian elimination is: it gives us an algorithm to write any matrix of the *general linear group*, $\mathrm{GL}(d, \mathcal{K})$, of size d over a field $\mathcal{K}$ as the product of elementary matrices and a diagonal matrix with all ones except one entry, using elementary operations. That entry in the diagonal is the determinant of the matrix. There are many ways to look at this phenomena. One simple way is: one can write the *matrix as a word in generators*. So in the language of computational group theory the word problem in $\mathrm{GL}(d, \mathcal{K})$ has an efficient algorithm – Gaussian elimination.

We write this paper to say that one can have a very similar result with split unitary groups as well. It is well known that unitary groups over a finite field are split. So we completely solve the problem for unitary groups over finite fields for most characteristics. However, over infinite fields, our algorithm works only for the split case. Split unitary groups are defined by the Hermitian form with maximal Witt

* *This work was supported by a SERB research grant MS: 831/13.*
Ayan Mahalanobis (Corresponding Author), Anupam Singh; IISER Pune, Dr. Homi Bhabha Road, Pashan, Pune 411008, India (email: ayan.mahalanobis@gmail.com, anupamk18@gmail.com).

index. From now on, by a **unitary group** we mean a **split unitary group**. We define **elementary matrices** and **elementary operations** for unitary groups. These matrices and operations are similar to that of *elementary transvections* and *elementary row-column operations* for special linear groups. Using these elementary matrices and elementary operations, we solve the word problem in unitary groups in a way that is very similar to the general linear groups. Similar algorithms are being developed for other classical groups and will be presented elsewhere.

Unitary groups are of interest in computational group theory, in the *matrix group recognition project*. In this paper, we work with a different set of generators than that is usual in computational group theory. The usual generators are called the *standard generators* [14, Tables 1 & 2]. Our generators, we call them elementary matrices and are defined later, have their root in the root spaces in Lie theory [6, Sections 11.3, 14.5] and have the disadvantage of being a larger set compared to that of the standard generators. However, standard generators being “multiplicative” in nature, depends on the primitive element of a finite field, works only for finite fields. On the other hand, our generators, work for arbitrary fields. Using standard generators, one needs to solve the discrete logarithm problem often. No such need arises in our case. In the current literature, the best row-column operations in unitary groups is by Costi [8] and implemented in Magma [3] by Costi and C. Schneider. Using their magma function *ClassicalRewriteNatural*, we show that our algorithm is much faster, see Figure 1. In Costi's algorithm one needs to compute various powers of ω a primitive element of the finite field. This makes his algorithm slower.

A need for row-column operations in classical groups was articulated by Seress [20, Page 677] in 1997. Computational group theory and in particular *constructive recognition of classical groups* have come a long way till then. We will not give a historical overview of this, an interested reader can find such an overview in the works of Brooksbank [5, Section 1.1], Leedham-Green and O'Brien [14, Section 1.3] and O'Brien [18]. Two recent works that are relevant to our work are Costi [8] and Ambrose et. al. [1]. Brooksbank [4, Section 5] deals with a similar algorithm which only works for finite fields.

In coming years, public key cryptography will go through a major change because of quantum computers. The ubiquitous public key cryptosystems like the ElGamal cryptosystem over elliptic curves and RSA will become obsolete. The need of the day are new public key cryptosystems whose security does not rely on the discrete logarithm problem or factoring integers. We study MOR cryptosystem on various groups with the hope to discover new quantum-secure cryptographic primitives.

In this paper, we only deal with unitary groups defined by the Hermitian form β defined later. The Hermitian form for the even-order case works for **all characteristic**. However, in the odd-order case the 2 in the upper-left makes it useless in the even characteristic. One can change this 2 to a 1 in β, however, then one needs to compensate that by putting $\frac{1}{2}$ in the generators. We tried, but were unable to extend our algorithm for the odd-order unitary group to even characteristic. For even-order unitary groups, the algorithm developed in this paper works for all characteristic. However, for the odd-order case only **odd characteristic will be considered**.

1.1. Notations

For the rest of the paper, let $\mathcal{K}$ be the quadratic extension of a field k with an automorphism $\sigma : x \mapsto \bar{x}$ of order two that fixes k elementwise. In the case of $\mathbb{C} : \mathbb{R}$, σ is the complex conjugation. In the case of a finite field $\mathbb{F}_{q^2} : \mathbb{F}_q$, σ is the map $x \mapsto x^q$. We define $\mathcal{K}^o = \{x \in \mathcal{K} \mid \bar{x} = -x\}$. We also denote $\mathcal{K}^1 = \{x \in \mathcal{K} \mid x\bar{x} = 1\}$. A $d \times d$ matrix X is called Hermitian (skew-Hermitian) if ${}^T\bar{X} = X$ (${}^T\bar{X} = -X$). Two important examples of $\mathcal{K} : k$ pairs that we have in mind for this work are $\mathbb{C} : \mathbb{R}$ and $\mathbb{F}_{q^2} : \mathbb{F}_q$.

The main result that we prove in this paper follows. The result is well known, however the algorithmic proof of the result is original. Moreover, this algorithm is of independent interest in other areas, for example, constructive recognition of classical groups. For a definition of elementary matrices and elementary operations, see Section 3.

Theorem A. *For $d \geq 4$, using elementary operations, one can write any matrix $\mathcal{A}$ in $U(d, \mathcal{K})$, the unitary group of size d over $\mathcal{K}$, as product of elementary matrices and a diagonal matrix. The diagonal* *matrix is of the following form:*

- $$\left(\begin{array}{cccc|cccc} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & \lambda & & & & \\ \hline & & & & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \\ & & & & & & & \bar{\lambda}^{-1} \end{array}\right)$$ *where* $\lambda\bar{\lambda}^{-1} = \det \mathcal{A}$ *and* $d = 2l$.

- $$\left(\begin{array}{c|cccc|cccc} \alpha & & & & & & & & \\ \hline & 1 & & & & & & & \\ & & \ddots & & & & & & \\ & & & 1 & & & & & \\ & & & & \lambda & & & & \\ \hline & & & & & 1 & & & \\ & & & & & & \ddots & & \\ & & & & & & & 1 & \\ & & & & & & & & \bar{\lambda}^{-1} \end{array}\right)$$ *where* $\alpha\bar{\alpha} = 1$ *and* $\alpha\lambda\bar{\lambda}^{-1} = \det \mathcal{A}$ *and* $d = 2l + 1$.

Here $\bar{\lambda}$ *is the image of* λ *under the automorphism* σ.

A trivial corollary (Theorem 6.1) of our algorithm is very similar to a result by Steinberg [21, §6.2], where he describes the generators of a projective-unitary group over odd characteristic. Our work is somewhat similar in nature to the work of Cohen et. al. [7], where the authors study generalized row-column operations in Chevalley groups. They did not study twisted groups.

We use the algorithm developed to construct a MOR cryptosystem in unitary groups and study its security.

2. Unitary groups

Let $\mathcal{K}$ be a field with a non-trivial field automorphism σ of order 2 with fixed field k. Let V be a vector space of dimension d over $\mathcal{K}$. We denote the image of α under σ by $\bar{\alpha}$. Let $\beta \colon V \times V \to \mathcal{K}$ be a non-degenerate Hermitian form, i.e., bar-linear in the first coordinate and linear in the second coordinate satisfying $\beta(x, y) = \overline{\beta(y, x)}$. We fix a basis for V and slightly abuse the notation to denote the matrix of β by β. Thus β is a non-singular matrix satisfying $\beta = {}^T\bar{\beta}$.

Definition 2.1 (Unitary Group). *The unitary group is:*

$$U(d, \mathcal{K}) = \{X \in GL(d, \mathcal{K}) \mid {}^T\bar{X}\beta X = \beta\}.$$

The special unitary group $SU(d, \mathcal{K})$ *consists of matrices of* $U(d, \mathcal{K})$ *of determinant* 1. *Note that the unitary group depends on the Hermitian form* β.

It is known that corresponding to equivalent Hermitian forms, corresponding unitary groups are conjugate in $\mathrm{GL}(d, \mathcal{K})$. However over a infinite field there could be more than one non-equivalent non-degenerate Hermitian form giving rise to more than one non-isomorphic unitary groups. In this article, we deal with a specific form β and the corresponding *split* unitary group. Recall, we assumed that characteristic of $\mathcal{K}$ is odd whenever d is odd. For the convenience of computations we index the basis of the vector space by $1, \ldots, l, -1, \ldots, -l$ when $d = 2l$ and by $0, 1, \ldots, l, -1, \ldots, -l$ when $d = 2l + 1$; where $l > 1$. We also fix the matrix β as follows:

- $d = 2l$ fix $\beta = \begin{pmatrix} & I_l \\ I_l & \end{pmatrix}$.
- $d = 2l + 1$ fix $\beta = \begin{pmatrix} 2 & & \\ & & I_l \\ & I_l & \end{pmatrix}$.

There are two important examples of fields: complex numbers $\mathbb{C}$ over reals $\mathbb{R}$ with σ the complex conjugation and the other, finite field $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$ with $\sigma : \alpha \mapsto \alpha^q$. In the case of $\mathbb{C} : \mathbb{R}$, Hermitian forms are classified by signatures and unitary groups denoted by $U(p, q)$ where $p + q = d$ (see [12] discussion following Theorem 6.19). The form corresponding to p being maximum is the split Hermitian form and is of interest in this paper. However there is only one non-degenerate Hermitian form up to equivalence [11, Corollary 10.4] over finite fields. In this case a unitary group will be denoted by $\mathrm{U}(d, q^2)$ and special unitary group as $\mathrm{SU}(d, q^2)$. A word of caution: in the literature $\mathrm{U}(d, q^2)$, $\mathrm{U}(d, \mathbb{F}_q)$ and $\mathrm{U}(d, q)$ are used interchangeably.

3. Elementary matrices and elementary operations in unitary groups

Solving the word problem in any group is of interest in computational group theory. In a special linear group, it can be easily solved using Gaussian elimination. However, for many groups, it is a very hard problem. In this paper we present a fast, cubic-time solution to the word problem in unitary groups.

Gaussian elimination in $\mathrm{SL}(d, \mathcal{K})$ uses elementary transvections as the elementary matrices and row-column operations as elementary operations. These elementary operations are multiplication by elementary matrices. The elementary matrices are of the form $I + te_{i,j}$ $(t \in \mathcal{K})$, where $e_{i,j}$ is the matrix unit with 1 in the $(i, j)^{\text{th}}$ position and zero elsewhere.

In the same spirit, one can define Chevalley-Steinberg generators for the unitary group [6, Section 14.5] as follows:

3.1. Elementary matrices for $\mathrm{U}(2l, \mathcal{K})$

In what follows, $l \geq 2$. For $1 \leq i, j \leq l$, $t \in \mathcal{K}$ and $s \in \mathcal{K}^o$:

$$\begin{aligned}
x_{i,j}(t) &= I + te_{i,j} - \bar{t}e_{-j,-i} \quad \text{for } i \neq j, \\
x_{i,-j}(t) &= I + te_{i,-j} - \bar{t}e_{j,-i} \quad \text{for } i < j, \\
x_{-i,j}(t) &= I + te_{-i,j} - \bar{t}e_{-j,i} \quad \text{for } i < j, \\
x_{i,-i}(s) &= I + se_{i,-i}, \\
x_{-i,i}(s) &= I + se_{-i,i},
\end{aligned}$$

3.2. Row-column operations for $\mathrm{U}(2l, \mathcal{K})$

Rephrasing the earlier definition in matrix format, we have three kinds of elementary matrices.

E1: $\begin{pmatrix} R & \\ & {}^T\bar{R}^{-1} \end{pmatrix}$ where $R = I + te_{i,j}$; $i \neq j$.

E2: $\begin{pmatrix} I & R \\ & I \end{pmatrix}$ where R is either $te_{i,j} - \bar{t}e_{j,i}$; $i < j$ or $se_{i,i}$.

E3: $\begin{pmatrix} I & \\ R & I \end{pmatrix}$ where R is either $te_{i,j} - \bar{t}e_{j,i}$; $i < j$ or $se_{i,i}$.

Let $g = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ be a $2l \times 2l$ matrix written in block form of size $l \times l$. Note the effect of multiplying g by matrices from above.

$$ER1: \begin{pmatrix} R & \\ & {}^T\bar{R}^{-1} \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} RA & RB \\ {}^T\bar{R}^{-1}C & {}^T\bar{R}^{-1}D \end{pmatrix}.$$
$$EC1: \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} R & \\ & {}^T\bar{R}^{-1} \end{pmatrix} = \begin{pmatrix} AR & B^T\bar{R}^{-1} \\ CR & D^T\bar{R}^{-1} \end{pmatrix}.$$

$$ER2: \begin{pmatrix} I & R \\ & I \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} A + RC & B + RD \\ C & D \end{pmatrix}.$$
$$EC2: \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} I & R \\ & I \end{pmatrix} = \begin{pmatrix} A & AR + B \\ C & CR + D \end{pmatrix}.$$

$$ER3: \begin{pmatrix} I & \\ R & I \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} A & B \\ RA + C & RB + D \end{pmatrix}.$$
$$EC3: \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} I & \\ R & I \end{pmatrix} = \begin{pmatrix} A + BR & B \\ C + DR & D \end{pmatrix}.$$

3.3. Elementary matrices for $\mathrm{U}(2l + 1, \mathcal{K})$

For $l \geq 2$, $1 \leq i, j \leq l$, $t \in \mathcal{K}$, $s \in \mathcal{K}^o$ and characteristic of $\mathcal{K}$ odd

$$\begin{aligned}
x_{i,j}(t) &= I + te_{i,j} - \bar{t}e_{-j,-i} && \text{for } i \neq j, \\
x_{i,-j}(t) &= I + te_{i,-j} - \bar{t}e_{j,-i} && \text{for } i < j, \\
x_{-i,j}(t) &= I + te_{-i,j} - \bar{t}e_{-j,i} && \text{for } i < j, \\
x_{i,-i}(s) &= I + se_{i,-i}, \\
x_{-i,i}(s) &= I + se_{-i,i}, \\
x_{i,0}(t) &= I - 2\bar{t}e_{i,0} + te_{0,-i} - t\bar{t}e_{i,-i}, \\
x_{0,i}(t) &= I + te_{0,i} - 2\bar{t}e_{-i,0} - t\bar{t}e_{-i,i},
\end{aligned}$$

3.4. Row-column operations for $\mathrm{U}(2l + 1, \mathcal{K})$

Rephrasing in matrix format:

E1: $\begin{pmatrix} 1 & & \\ & R & \\ & & {}^T\bar{R}^{-1} \end{pmatrix}$ where $R = I + te_{i,j}$; $i \neq j$.

E2: $\begin{pmatrix} 1 & & \\ & I & R \\ & & I \end{pmatrix}$ where R is either $te_{i,j} - \bar{t}e_{j,i}$; $i < j$ or $se_{i,i}$.

E3: $\begin{pmatrix} 1 & & \\ & I & \\ & R & I \end{pmatrix}$ where R is either $te_{i,j} - \bar{t}e_{j,i}$; $i < j$ or $se_{i,i}$.

E4: $\begin{cases} \begin{pmatrix} 1 & & R \\ -2\bar{R} & I & -{}^TR\bar{R} \\ & & I \end{pmatrix} & \text{where } R = te_i \\ \begin{pmatrix} 1 & R & \\ & I & \\ -2\bar{R} & -{}^TR\bar{R} & I \end{pmatrix} & \text{where } R = te_i \end{cases}$

Here e_i is the row vector with 1 at i^{th} place and zero elsewhere. Let $g = \begin{pmatrix} \alpha & X & Y \\ E & A & B \\ F & C & D \end{pmatrix}$ be a $(2l+1)\times(2l+1)$ matrix where A, B, C, D are $l \times l$ matrices. The matrices $X = (X_1, X_2, \ldots, X_l)$, $Y = (Y_1, Y_2, \ldots, Y_l)$, $E = {}^T(E_1, E_2, \ldots, E_l)$ and $F = {}^T(F_1, F_2, \ldots, F_l)$ are rows of length l. Furthermore $\alpha \in \mathcal{K}$. Note the effect of multiplication by elementary matrices from above is as follows:

$$ER1: \begin{pmatrix} 1 & & \\ & R & \\ & & {}^T\bar{R}^{-1} \end{pmatrix} \begin{pmatrix} \alpha & X & Y \\ E & A & B \\ F & C & D \end{pmatrix} = \begin{pmatrix} \alpha & X & Y \\ RE & RA & RB \\ {}^T\bar{R}^{-1}F & {}^T\bar{R}^{-1}C & {}^T\bar{R}^{-1}D \end{pmatrix}.$$

$$EC1: \begin{pmatrix} \alpha & X & Y \\ E & A & B \\ F & C & D \end{pmatrix} \begin{pmatrix} 1 & & \\ & R & \\ & & {}^T\bar{R}^{-1} \end{pmatrix} = \begin{pmatrix} \alpha & XR & Y{}^T\bar{R}^{-1} \\ E & AR & B{}^T\bar{R}^{-1} \\ F & CR & D{}^T\bar{R}^{-1} \end{pmatrix}.$$

$$ER2: \begin{pmatrix} 1 & & \\ & I & R \\ & & I \end{pmatrix} \begin{pmatrix} \alpha & X & Y \\ E & A & B \\ F & C & D \end{pmatrix} = \begin{pmatrix} \alpha & X & Y \\ E+RF & A+RC & B+RD \\ F & C & D \end{pmatrix}.$$

$$EC2: \begin{pmatrix} \alpha & X & Y \\ E & A & B \\ F & C & D \end{pmatrix} \begin{pmatrix} 1 & & \\ & I & R \\ & & I \end{pmatrix} = \begin{pmatrix} \alpha & X & XR+Y \\ E & A & AR+B \\ F & C & CR+D \end{pmatrix}.$$

$$ER3: \begin{pmatrix} 1 & & \\ & I & \\ & R & I \end{pmatrix} \begin{pmatrix} \alpha & X & Y \\ E & A & B \\ F & C & D \end{pmatrix} = \begin{pmatrix} \alpha & X & Y \\ E & A & B \\ RE+F & RA+C & RB+D \end{pmatrix}.$$

$$EC3: \begin{pmatrix} \alpha & X & Y \\ E & A & B \\ F & C & D \end{pmatrix} \begin{pmatrix} 1 & & \\ & I & \\ & R & I \end{pmatrix} = \begin{pmatrix} \alpha & X+YR & Y \\ E & A+BR & B \\ F & C+DR & D \end{pmatrix}.$$

For E4 we only write the equations that we need later.

- Let the matrix g has $C = \mathrm{diag}(d_1, \ldots, d_l)$.

$$ER4: \; [(I + te_{0,-i} - 2\bar{t}e_{i,0} - t\bar{t}e_{i,-i})g]_{0,i} = X_i + td_i$$

$$EC4: \; [g(I + te_{0,-i} - 2\bar{t}e_{i,0} - t\bar{t}e_{i,-i})]_{-i,0} = F_i - 2\bar{t}d_i.$$

- Let the matrix g has $A = \mathrm{diag}(d_1, \ldots, d_l)$.

$$ER4: \; [(I + te_{0,i} - 2\bar{t}e_{-i,0} - t\bar{t}e_{-i,i})g]_{0,i} = X_i + td_i$$

$$EC4: \; [g(I + te_{0,-i} - 2\bar{t}e_{i,0} - t\bar{t}e_{i,-i})]_{i,0} = E_i - 2\bar{t}d_i.$$

3.5. Row-interchange matrices

We need certain row interchange matrices, multiplication with these matrices from left, interchanges i^{th} row with $-i^{th}$ row for $1 \leq i \leq l$. These are certain Weyl group elements. These matrices can be produced as follows: for $s \in \mathcal{K}^o$,

$$w_{i,-i}(s) = x_{i,-i}(s)x_{-i,i}(-s^{-1})x_{i,-i}(s) = I + se_{i,-i} - e_{i,i} - s^{-1}e_{-i,i} - e_{-i,-i}.$$

Note that our row interchange multiplies one row by s and the other by $-s^{-1}$ and then swaps them. This scalar multiplication of rows produce no problem for our cause.

4. Gaussian elimination in unitary group

Now we present the main result of this paper, two algorithms, one for even-order unitary groups and other for the odd-order unitary groups.

4.1. The algorithm for even-order unitary groups

Let $g = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ be an element of the unitary group $U(2l, \mathcal{K})$. One principal reason our algorithm works is that we are able to exploit a symmetry that comes out of the use of the Hermitian form β described earlier.

Notice that, ${}^T\bar{g}\beta g = \beta$ implies after straightforward computations that ${}^T\bar{C}A + {}^T\bar{A}C = 0$ and ${}^T\bar{D}B + {}^T\bar{B}D = 0$. This implies that ${}^T\bar{A}C$ and ${}^T\bar{B}D$ are skew-Hermitian matrices. We now describe the algorithm.

Step 1 Using ER1 and EC1 make A into a diagonal matrix. This is the usual Gaussian elimination algorithm. This new diagonal matrix will be referred to A as well. There are two possibilities.

a The diagonal matrix has full rank and is of the form $\mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_l)$, where each λ_i are non-zero.

b The diagonal matrix A do not have a full rank and is of rank r less than l and is of the form $\mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r, 0, 0 \ldots, 0)$.

Step 2 In case a, use ER3 to make C into a zero matrix. In case b, bottom $l - r$ rows in A are zero. Make the top r rows of C zero using ER3. We now interchange bottom $l - r$ rows of A with the corresponding $l - r$ rows of C. This makes C a zero matrix. We claim that A must be of full rank. We know that ${}^T\bar{C}B + {}^T\bar{A}D = I$. Since, $C = 0$, A must be of full rank. We now go back to Step 1 and make A a diagonal matrix.

Step 3 Note that ${}^T\bar{D}A + {}^T\bar{B}C = I$ and since C is zero and A is diagonal makes D a diagonal matrix of full rank l. Using this diagonal matrix, we can make B a zero matrix using ER2. So now we have a diagonal matrix instead of g.

Step 4 In this step we only work with A. Using ER1 and EC1, we reduce all diagonal elements of A to 1 except the last.

4.2. The algorithm for odd-order unitary groups

Recall that for odd-order unitary groups, we assumed the characteristic of $\mathcal{K}$ to be odd. Let $g = \begin{pmatrix} \alpha & X & Y \\ E & A & B \\ F & C & D \end{pmatrix}$ be an element of the unitary group $U(2l+1, \mathcal{K})$. As with even-order case, our algorithm exploits the symmetry that come out of the chosen bilinear form β. From the equation ${}^T\bar{g}\beta g = \beta$ we get two useful equations:

$$2{}^T\bar{X}X + {}^T\bar{C}A + {}^T\bar{A}C = 0 \quad (1)$$
$$2{}^T\bar{Y}Y + {}^T\bar{D}B + {}^T\bar{B}D = 0 \quad (2)$$

Now if $X = 0$, ${}^T\bar{A}C$ is skew-Hermitian and similarly for the other case ${}^T\bar{B}D$ is skew-Hermitian. The reader will notice our insistence in making X zero as quickly as possible in the algorithm. Once we do that rest of the algorithm is very similar to that of the even-order algorithm.

The algorithm is as follows:

Step 1 Using ER1 and EC1, make A a diagonal matrix. This new diagonal matrix will be referred to A as well. Two things can happen

- **a** The diagonal matrix A has full rank and is of the form $\operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_l)$ where λ_i are non-zero for $1 \le i \le l$.
- **b** The matrix A is not of full rank, but of rank r and is $\operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r, 0, \ldots, 0)$ where λ_i are non-zero for $1 \le i \le r$.

Step 2 The purpose of this step is to make X and E zero using A. In the case a above, this can be easily done using ER4 and EC4 respectively. In the case b above, from Lemma 5.1, if $x_1 = x_2 = \ldots = x_r = 0$ then $X = 0$. However $x_1, x_2, \ldots, x_r$ can be made zero as above and that will make whole of X zero. Similarly, use the non-zero diagonals of A to make the corresponding entries of E zero.

Step 3 The purpose of this step is to make C zero matrix. We first use the non-zero diagonal entries in A to make the corresponding rows in C zero. If there are any zero rows in A, i.e., we are in case b of Step 1, we use the row interchange operation to exchange the zero rows of A with the corresponding ones in C. Then C is a zero matrix and that makes A of full rank. Diagonalize A using ER1 and EC1 and make E a zero matrix. This makes D a full-rank diagonal matrix.

Step 4 Using ER2 and D we make B a zero matrix.

Step 5 We now have a diagonal matrix, using ER1 and EC1 we can make the diagonal entries of A except the last one 1.

5. Some lemmas

Lemma 5.1. *Let g be an element of $U(2l+1, \mathcal{K})$ as described earlier. Furthermore assume that A is of the form $\operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r, 0, \ldots, 0)$ where λ_i are non-zero and the row vector X has first r entries 0. Then X is zero.*

Proof. Notice that from the equation ${}^T\bar{g}\beta g = \beta$, it follows that $2{}^T\bar{X}X + {}^T\bar{C}A + {}^T\bar{A}C = 0$. Then ${}^T\bar{X}X$ is a matrix with the lower right $(l-r) \times (l-r)$ block possibly non-zero. However since, A is a diagonal matrix with lower $l-r$ entries zero, it is clear from the equation that the possible non-zero block is zero and this proves that $X = 0$. □

Lemma 5.2. *Once E, C and X are zero in g defined earlier, it follows that F and Y are zero.*

Proof. The important equation from ${}^T\bar{g}\beta g = \beta$ for this lemma is $2{}^T\bar{X}\alpha + {}^T\bar{C}E + {}^T\bar{A}F = 0$ from which it follows that $F = 0$. Then $Y = 0$ follows from $2\bar{\alpha}Y + {}^T\bar{F}B + \bar{E}D = 0$ □

A cautious reader must have noticed that in the definition of the generators we define $x_{i,-i}(s) = I + se_{i,-i}$ and $x_{-i,i}(s) = I + se_{-i,i}$ where $s \in \mathcal{K}^o$. We need these generators to clear diagonal elements from C and B respectively. We only talk about clearing diagonal elements from C, B follows similarly. Now if A is diagonal and λ_i is a non-zero element in the diagonal, then using $RA + C$ we will clear $c_{i,i}$ in C. Notice that from earlier discussion it follows $\bar{\lambda}_i c_{i,i} = -\bar{c}_{i,i}\lambda_i$. Then take $s = -\frac{c_{i,i}}{\lambda}$ and it is easy to check that it belongs to $\mathcal{K}^o$.

5.1. Proof of Theorem A

Proof. Let $g \in \mathrm{U}(d, \mathcal{K})$. Using the Gaussian elimination above we can reduce g to a matrix of the form $\mathrm{diag}(1, \ldots, 1, \lambda, 1, \ldots, 1, \bar{\lambda}^{-1})$ when $d = 2l$ and $\mathrm{diag}(\alpha, 1 \ldots, 1, \lambda, 1, \ldots, 1, \bar{\lambda}^{-1})$ when $d = 2l + 1$. We further note that row-column operations are multiplication by elementary matrices from left or right and each of these elementary matrices have determinant one. Thus we get the required result. □

5.2. Asymptotic complexity is $O(l^3)$

In this section, we show that the asymptotic complexity of the algorithm that we developed is $O(l^3)$. We count the number of field multiplications. We can break the algorithm into three parts. One, reduce A to a diagonal, then deal with C and then with D. It is easy to see that reducing A to the diagonal has complexity $O(l^3)$ and dealing with C and D has complexity $O(l^3)$. Row interchange has complexity $O(l^2)$. In the odd-order case there is a complexity of $O(l)$ to deal with X, Y, E and F. In all, the worst case complexity is $O(l^3)$.

6. Finite unitary groups

In the next section, we talk about cryptography. In cryptography, we need to deal explicitly with finite fields. In this context, when $\mathcal{K} = \mathbb{F}_{q^2}$, we prove a theorem similar in spirit to Steinberg [21, Section 6.2]. The proof is an obvious corollary to our algorithm.

Theorem 6.1. *Fix an element ζ which generates the cyclic group $\mathbb{F}_{q^2}^{\times}$, the subgroup $\mathbb{F}_{q^2}^{1}$ is generated by $\zeta_1 = \zeta^{q-1}$. We add following matrices to the respective set of elementary matrices:*

- $h(\zeta) = \mathrm{diag}(1, \ldots, \zeta, 1, \ldots, \bar{\zeta}^{-1})$ *whenever* $d = 2l$.
- $\begin{cases} h(\zeta) = & \mathrm{diag}(1, \ldots, \zeta, 1, \ldots, \bar{\zeta}^{-1}) \\ h(\zeta_1) = & \mathrm{diag}(\zeta_1, 1, \ldots, 1) \end{cases}$ *whenever* $d = 2l + 1$.

Then the group $U(d, q^2)$ is generated by elementary matrices and the matrices defined above.

6.1. Special unitary group $\mathrm{SU}(d, q^2)$

In the case of $\mathrm{SU}(2l, q^2)$ a simple and straightforward enhancement of our algorithm reduces a matrix $g \in \mathrm{SU}(2l, q^2)$ to the identity matrix. Thus the word problem in $\mathrm{SU}(2l, q^2)$ is completely solved as with $\mathrm{SL}(d, q)$ using only elementary matrices; this is particularly useful for the MOR cryptosystem. An analysis of a MOR cryptosystem similar to the MOR cryptosystem over $\mathrm{SL}(d, q)$ [15] will be done in the next section.

For the reduction to identity, note that Theorem 1.1 reduces g to $\mathrm{diag}(1, \ldots, 1, \lambda, 1 \ldots, 1, \bar{\lambda}^{-1})$. However, since $\det(g) = \lambda\bar{\lambda}^{-1} = 1$, we have $\lambda = \bar{\lambda}$ and $\lambda \in \mathbb{F}_q^{\times}$. Thus the word problem is completely solved for even characteristics. For odd characteristics, let $s = \varepsilon\lambda$ where $\varepsilon \in \mathcal{K}^o$, then

$$w_{l,-l}(s)w_{l,-l}(-\varepsilon) = \mathrm{diag}(1, \ldots, 1, \lambda, 1, \ldots, \lambda^{-1}).$$

So if we add $w_{l,-l}(s)w_{l,-l}(-\varepsilon)$ to the output of Theorem 1.1, we have the identity matrix.

In the case of $\mathrm{SU}(2l + 1, q^2)$ we need to add an extra generator $h(\zeta_1) = \mathrm{diag}(\zeta_1, 1, \ldots, 1)$ where ζ_1 is a generator of $\mathbb{F}^1_{q^2}$. Now we can reduce an element of the form $\mathrm{diag}(\alpha, 1 \ldots, \lambda, 1, \ldots, \bar{\lambda}^{-1})$ to $\mathrm{diag}(1, 1 \ldots, \lambda, 1, \ldots, \bar{\lambda}^{-1})$ by multiplying with the suitable power of $h(\zeta_1)$. Note that finding the suitable power involves solving a discrete logarithm problem. Then we use similar computations for even-order case to reduce $\mathrm{diag}(1, 1 \ldots, \lambda, 1, \ldots, \bar{\lambda}^{-1})$ to identity.

7. The MOR cryptosystem on unitary groups

In this section, we will work with the MOR cryptosystem over $U(2l, q^2)$ most of time. However, we will occasionally refer to the odd-order unitary group as well. Briefly speaking, the MOR cryptosystem is a simple and straightforward generalization of the classic ElGamal cryptosystem and was put forward by Paeng et. al. [19]. In a MOR cryptosystem one works with the automorphism group rather than the group itself. It provides an interesting change in perspective in public-key cryptography – from finite cyclic groups to finite non-abelian groups. The MOR cryptosystem was studied for the special linear group in details by Mahalanobis [15]. For many other classical groups, except the orthogonal groups, the analysis of a MOR cryptosystem remains almost the same. So we will remain brief in this paper and refer an interested reader to [15] (see also [16]).

The description of the MOR cryptosystem is as follows:

Let $G = \langle g_1, g_2, \ldots, g_s\rangle$ be a finite group. Let ϕ be a non-identity automorphism.

- **Public-key:** Let $\{\phi(g_i)\}_{i=1}^s$ and $\{\phi^{\mathfrak{m}}(g_i)\}_{i=1}^s$ is public.
- **Private-key:** The integer $\mathfrak{m}$ is private.

Encryption:
To encrypt a plaintext $\mathfrak{M} \in G$, get an arbitrary integer $r \in [1, |\phi|]$ compute ϕ^r and $\phi^{r\mathfrak{m}}$. The ciphertext is $(\phi^r, \phi^{r\mathfrak{m}}(\mathfrak{M}))$.
Decryption:
After receiving the ciphertext $(\phi^r, \phi^{r\mathfrak{m}}(\mathfrak{M}))$, the user knows the private key $\mathfrak{m}$. So she computes $\phi^{\mathfrak{m}r}$ from ϕ^r and then computes $\mathfrak{M}$.

To develop a MOR cryptosystem we need a thorough understanding of the automorphisms group of the group involved. The automorphisms of unitary groups are well described in the literature. We mention them briefly to facilitate further discussion.

7.1. Automorphism group of unitary groups

First we define the similitude group. We need these groups to define diagonal automorphisms.

Definition 7.1 (Unitary similitude group)**.** *The unitary similitude group is defined as:*

$$GU(d,q^2) = \{X \in GL(d,q^2) \mid {}^T\bar{X}\beta X = \mu\beta, \textit{for some } \mu \in \mathbb{F}_q^{\times}\}.$$

Note that the multiplier μ defines a group homomorphism from $\mathrm{GU}(d,q^2)$ to $\mathbb{F}_q^{\times}$ with kernel the unitary group.

Conjugation Automorphisms: When the conjugation map $g \mapsto ngn^{-1}$ for $n \in \mathrm{GU}(d,q^2)$ is an automorphism, we call it a conjugation automorphism. They are a composition of two types of automorphisms – inner automorphisms given as conjugation by elements of $\mathrm{U}(d,q^2)$ and diagonal automorphisms given as conjugation by diagonals of $\mathrm{GU}(d,q^2)$.

Central Automorphisms: Let $\chi\colon \mathrm{U}(d,q^2) \to \mathbb{F}_{q^2}^1$ be a group homomorphism. Then the central automorphism c_χ is given by $g \mapsto \chi(g)g$. Since $[\mathrm{U}(d,q^2),\mathrm{U}(d,q^2)] = [\mathrm{SU}(d,q^2),\mathrm{SU}(d,q^2)] = \mathrm{SU}(d,q^2)$ [11, Theorem 11.22], any χ is equivalent to a group homomorphism from $\mathrm{U}(d,q^2)/\mathrm{SU}(d,q^2)$ to $\mathbb{F}_{q^2}^1$. There are at most $q+1$ such maps.

Field Automorphisms: For any automorphism σ of the field $\mathbb{F}_{q^2}$, replacing all entries of a matrix by their image under σ give us a field automorphism.

The following theorem, due to Dieudonné [9, Theorem 25], describes all automorphisms:

Theorem 7.2. *Let q be odd and $d \geq 4$. Then any automorphism ϕ of the unitary group $U(d,q^2)$ is written as $c_\chi\iota\delta\sigma$ where c_χ is a central automorphism, ι is an inner automorphism, δ is a diagonal automorphism and σ is a field automorphism.*

As we saw above there are three kind of automorphisms in an unitary group. One is conjugation automorphism, the others are central and field automorphisms. A central automorphism being multiplication by an element of the center which is a field element. Exponentiation of a central automorphism will give rise to a discrete logarithm problem in $\mathbb{F}_{q^2}$. Similar is the case with a field automorphism. So the only choice for a better MOR cryptosystem is a conjugation automorphism.

Once, we have decided that the automorphism that we are going to use in the MOR cryptosystem will act by conjugation. Further analysis is straightforward and follows [15, Section 7]. Recall that we insisted that automorphisms in the MOR cryptosystem are presented as action on generators. In this case, the generators are *elementary matrices* and the group is a *special unitary group* of even-order. Other groups can be used and analyzed similarly. Note that two things can happen: one can find the conjugator element for the automorphism in use, finding the conjugator up to a scalar multiple is enough or one cannot find the conjugator in the automorphism.

In the first case, the discrete logarithm problem in the automorphism becomes a discrete logarithm problem in a matrix group. Assume that we found the conjugating matrix A up to a scalar multiple, where $A \in \mathrm{GU}(d,q^2)$. Now the discrete logarithm problem in ϕ becomes a discrete logarithm problem in A. One can show that by suitably choosing A, the discrete logarithm in A is embedded in the field $\mathbb{F}_{q^{2d}}$. This argument is presented in details [15, Section 7.1]. We will not repeat it here. In the next section (reduction of security), we show that one can find this conjugating element for unitary groups.

The success of any cryptosystem comes from a balance between speed and security. In this paper, we deal with both speed and security of the MOR cryptosystem briefly. For an implementation of the MOR cryptosystem, we need to compute power of an automorphism. The algorithm of our choice is the famous **square-and-multiply** algorithm. Since we do not use any special algorithm for squaring, squaring and multiplying is the same for us. So we talk about multiplying two automorphisms. We present automorphisms as action on generators, i.e., $\phi(g_i)$ is a matrix for $i = 1,2,\ldots,s$. The first step of the algorithm is to find the word in generators from the matrix[1]. So now the automorphism is $\phi(g_i) = w_i$ where each w_i is a word in generators. Once that is done then composing with an automorphism is

[1] *One can also present the automorphisms as word in generators, we choose matrices.*

substituting each generator in the word by another word. This can be done fast. The challenging thing is to find the matrix corresponding to the word thus formed. This is not a hard problem, but can be both time and memory intensive. What is the best way to do it is still an open question! However, there are many shortcuts available. One being an obvious time-memory trade off, like storing matrices corresponding to a word in generators. The other being there are many trivial and non-trivial relations among these generators and moreover these generators are sparse matrices. One can use these properties in the implementation.

This problem, which is of independent interest in computational group theory and is a reason that we insist on automorphisms being presented as generators for the MOR cryptosystem. For more information, see [15, Section 8].

7.2. Reduction of security

In this subsection, we show that for unitary groups, the security of the MOR cryptosystem reduces to the discrete logarithm problem in $\mathbb{F}_{q^{2d}}$. This is the same as saying that we can find the conjugating matrix up to a scalar multiple. Let ϕ be an automorphism that works by conjugation, i.e., $\phi = \iota_g$ for some g and we try to determine g.

Step 1: The automorphism ϕ is presented as action on generators. Thus $\phi(x_{i,-i}(s)) = g(I + se_{i,-i})g^{-1} = I + sge_{i,-i}g^{-1}$. This implies that we know $\varepsilon ge_{i,-i}g^{-1}$ and similarly $\varepsilon ge_{-i,i}g^{-1}$ for a fixed $\varepsilon \in \mathcal{K}^o$. We first claim that we can determine $N := gD$ where D is diagonal.

When $d = 2l$, write g in the column form $[G_1, \ldots G_l, G_{-1}, \ldots, G_{-l}]$. Now,

1. $[G_1, \ldots G_l, G_{-1}, \ldots, G_{-l}]\,\varepsilon e_{i,-i} = [0, \ldots, 0, \varepsilon G_i, 0, \ldots, 0]$ where G_i is at $-i$th place. Multiplying this with g^{-1} gives us scalar multiple of G_i, say d_i.

2. $[G_1, \ldots G_l, G_{-1}, \ldots, G_{-l}]\,\varepsilon e_{-i,i} = [0, \ldots, 0, \varepsilon G_{-i}, 0, \ldots, 0]$ where G_{-i} is at ith place. Multiplying this with g^{-1} gives us scalar multiple of G_{-i}, say d_{-i}.

Thus we get $N = gD$ where D is a diagonal matrix $\operatorname{diag}(d_1, \ldots, d_l, d_{-1}, \ldots, d_{-l})$. In the case when $d = 2l + 1$ we write $g = [G_0, G_1, \ldots, G_l, G_{-1}, \ldots, G_{-l}]$ and get scalar multiple of columns G_i and G_{-i}. We now use $x_{i,0}(t)$ and $x_{0,i}(t)$ to get linear combination of G_0 with G_i or G_{-i}, say we get $\alpha G_0 + \beta G_{-1}$. In this case we get $N = gD$ where D is of the form

$$\begin{pmatrix} \alpha & & & & & & \\ & d_1 & & & & & \\ & & \ddots & & & & \\ & & & d_l & & & \\ \beta & & & & d_{-1} & & \\ & & & & & \ddots & \\ & & & & & & d_{-l} \end{pmatrix}.$$

Step 2: Now we compute $N^{-1}\phi(x_r(t))N = D^{-1}g^{-1}(gx_r(t)g^{-1})gD = D^{-1}x_r(t)D$. Substituting various $x_r(t)$ it amounts to computing $D^{-1}e_rD$. When $d = 2l$, we first compute $D^{-1}(e_{i,j} - e_{-j,-i})D$ and get $d_i^{-1}d_j$, $d_{-i}^{-1}d_{-j}$ for $i \neq j$. Then we compute $D^{-1}e_{i,-i}D, D^{-1}e_{-i,i}D$ and get $d_i d_{-i}^{-1}, d_{-i}d_i^{-1}$. We form a matrix

$$\operatorname{diag}(1, d_2^{-1}d_1, \ldots, d_l^{-1}d_1, d_{-1}^{-1}d_1, \ldots, d_{-l}^{-1}d_1)$$

and multiply it to $N = gD$ to get $d_1 g$. Thus we can determine g up to a scalar multiple and the attack follows [15, Section 7.1.1].

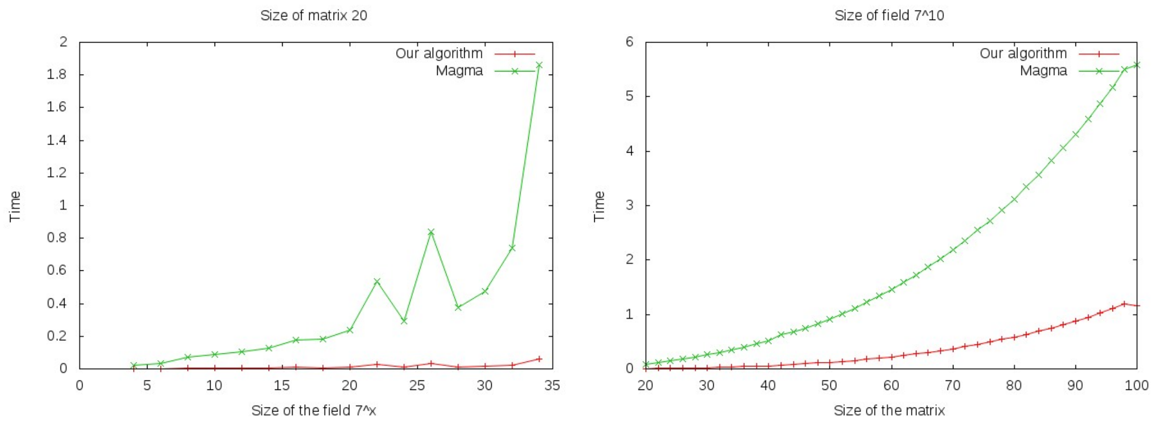

Figure 1. Some simulations comparing our algorithm with the one inbuilt in Magma.

In the case $d = 2l+1$, the matrix D is almost a diagonal matrix except the first column. However while computing $D^{-1}(e_{12} - e_{-2,-1})D$ we also get $d_{-2}^{-1}\beta$ and by computing $D^{-1}(e_{0,1} - 2e_{-1,0} - e_{-1,1})D$ we get $\alpha^{-1}D$. Thus we can multiply $\alpha G_0 + \beta G_{-1}$ by $\beta^{-1}d_1 = \beta^{-1}d_{-2}d_{-2}^{-1}d_{-1}d_{-1}^{-1}d_1$ and get $\alpha\beta^{-1}d_1G_0 + d_1G_{-1}$. With the computation in even case we can determine d_1G_{-1} and hence can determine αG_0. Furthermore, since we know $\alpha^{-1}d_1$ we can determine d_1G_0 thus in this case as well we can determine d_1g, i.e., g up to a scalar multiple.

8. Conclusion

For us, this paper is an interplay of finite (non-abelian) groups and public key cryptography. Computational group theory, in particular computations with quasi-simple groups have a long and distinguished history [2, 7, 13, 14, 17]. The interesting thing to us is, some of the questions that arise naturally when dealing with the MOR cryptosystem are interesting in its own right in computational group theory and are actively studied. The row-column operations that we developed is one example of that. In the row-column operations we developed, we used a different set of generators. These generators have a long history starting with Chevalley. In our knowledge, we are the first to use them in row-column operations in unitary groups. Earlier works were mostly done using the standard generators. It seems that Chevalley generators might offer a paradigm shift in algorithms with quasi-simple groups. In Magma, there is an implementation of row-column operations in unitary groups in a function *ClassicalRewriteNatural*. We compared that function with our algorithm in an actual implementation on even order unitary groups using identical parameters. To select parameters for our simulation, we followed Costi's work [8, Table 6.2]. In one case, the characteristic of the field was fixed at 7 and the size of the matrix at 20, we varied the degree of the extension of the field from 4 to 34. We then picked at random elements from the GeneralUnitaryGroup and timed our algorithm. We did the same with the magma function using special unitary group. The final time was the average over one thousand such repetitions. Times of both these algorithms were tabulated and is presented in Figure 1. In another case, we kept the field fixed at 7^{10} and changed the size of the matrix. In all cases, the final time was the average of one thousand random repetitions. The timing was tabulated and presented in Figure 1. It seems the our algorithm is much better than that of Costi's from all aspects.

Acknowledgment: Authors owe a debt of gratitude to the referee and the handling editor for careful reading and wonderful comments which has improved the paper substantially.

References

[1] S. Ambrose, S. Murray, C. E. Praeger, C. Schneider, Constructive membership testing in black–box classical groups, Proceedings of The Third International Congress on Mathematical Software, LNCS 6327 (2011) 54–57.

[2] H. Bäärnhielm, D. Holt, C. R. Leedham-Green, E. A. O'Brien, A practical model for computation with matrix groups, J. Symb. Comput. 68 (2015) 27–60.

[3] W. Bosma, J. Cannon, C. Playoust, The Magma algebra system. I: The user language, J. Symb. Comput. 24(3-4) (1997) 235–265.

[4] P. Brooksbank, Constructive recognition of classical groups in their natural representation, J. Symb. Comput. 35(2) (2003) 195–239.

[5] P. Brooksbank, Fast constructive recognition of black–box unitary groups, LMS J. Comput. Math. 6 (2003) 162–197.

[6] R. Carter, Simple Groups of Lie Type, New York: John Wiley and Sons, 1972.

[7] A. M. Cohen, S. H. Murray, D. E. Taylor, Computing in groups of Lie type, Math. Comput. 73 (2004) 1477–1498.

[8] E. Costi, Constructive Membership Testing in Classical Groups, Ph.D. thesis, Queen Mary, Univ. of London, 2009.

[9] J. Dieudonne, On the automorphisms of the classical groups. with a supplement by Loo-Keng Hua, Mem. Amer. Math. Soc. 2 (1951) vi+122.

[10] J. F. Grcar, Mathematicians of Gaussian elimination, Notices Amer. Math. Soc. 58(6) (2011) 782–792.

[11] L. C. Grove, Classical Groups and Geometric Algebra, vol. 39, American Mathematical Society, Graduate Studies in Mathematics, 2002.

[12] N. Jacobson, Basic Algebra I, W. H. Freeman and Company, New York, 1985.

[13] W. Kantor À. Seress, Black box classical groups, vol. 149, Mem. Amer. Math. Soc. 149 (2001) viii+168.

[14] C. R. Leedham–Green, E. A. O'Brien, Constructive recognition of classical groups in odd characteristic, J. Algebra 322(3) (2009) 833–881.

[15] A. Mahalanobis, A simple generalization of the ElGamal cryptosystem to non–abelian groups II, Commun. Algebra 40(9) (2012) 3583–3596.

[16] C. Monico, Cryptanalysis of matrix–based MOR system, Commun. Algebra 44(1) (2016) 218–227.

[17] A. C. Niemeyer, C. E. Praeger, A recognition algorithm for classical groups over finite fields, Proc. London Math. Soc. 77(1) (1998) 117–169.

[18] E. A. O'Brien, Algorithms for matrix groups, Groups St. Andrews 2009 in Bath, II, London Math. Soc. Lecture Note Ser., 388 (Cambridge Univ. Press, Cambridge, 2011), 297–323.

[19] SH. Paeng, KC. Ha, J. H. Kim, S. Chee, C. Park, New public key cryptosystem using finite non–abelian groups, Crypto 2001 (J. Kilian, ed.), LNCS, vol. 2139, Springer–Verlag, (2001) 470–485.

[20] Á. Seress, An introduction to computational group theory, Notices Amer. Math. Soc. 44(6) (1997) 671–679.

[21] R. Steinberg, Variations on a theme of Chevalley, Pacific J. Math. 9 (1959) 875–891.