# Recognition of Finite Black Box Groups

A dissertation presented

by

**Sergey Bratus**

to

The Department of Mathematics

In partial fulfillment of the requirements

for the Degree of

**Doctor of Philosophy**

in the field of

Mathematics

**Northeastern University**

**Boston, Massachusetts**

March 1999

**Recognition of Finite Black Box Groups**

**Approved by**
**Thesis Committee**

To my wife Anna

# Recognition of Finite Black Box Groups

by

Sergey L. Bratus

ABSTRACT OF DISSERTATION

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy in Mathematics

in the Graduate School of Arts and Sciences of

Northeastern University, March 1999

In this thesis we develop and apply the constructive approach to recognition of black box groups isomorphic to the symmetric and alternating groups, and the special linear and projective special linear groups over finite fields. This approach is based on the work of Cooperman, Finkelstein and Linton [CFL], and a joint work with Pak [BP].

Polynomial time recognition algorithms are presented for constructing an isomorphism between a *black box* group $G$ known to be isomorphic to one of $S_n$, $A_n$, $SL(n, q)$, or a central extension of $PSL(n, q)$. The algorithm is based on recognizing the action of $G$ on a simple conjugacy class $\mathcal{C}$ (transpositions, 3-cycles, transvections) that generates $G$.

Consider a finite group $G$ which is known to be isomorphic to one of the classical groups mentioned above, but is defined in a way such that the relationship of $G$ to its natural representation $\Gamma$ is not transparent. The presentation of $G$ is assumed to allow efficient computation of products and inverses of elements of $G$, as well as efficient testing of whether a given element $x \in G$ is the identity. For example, $G$ may be given as a subgroup of $S_N$, $N \gg n$, defined by a set of generators $S \subset S_N$, or $G$ may be defined as a subgroup generated by a set of matrices in a much larger matrix group $GL(N, q')$, possibly over a different finite field. The notion of a *black box* group was introduced by Babai and Szemerédi to provide a theoretical abstraction for such cases when the computation can rely mainly on the group structure (the raw multiplication table) of $G$. This abstract model addresses situations when using the actual form in which elements of $G$ are represented is either not very helpful, or not possible.

In the *black box* computational model the elements of a finite group $G$ are encoded by bit strings (of uniform length $M$ bits) that completely hide their actual identity (with respect to the natural representation $\Gamma$), while the group multiplication and other operations are performed by an oracle, the *black box*. The oracle inputs bit strings representing the group elements and provides the the product of two group elements (as a new bit string), the inverse of a group element (as a new bit string), or tests if the input group element (possibly obtained as a result of a series of such group operations) is the group's identity. Constructing an explicit isomorphism $\Theta : G \to \Gamma$ to the natural representation $\Gamma$ of $G$ to recover the actual identity of $G$'s elements is the purpose of a *constructive recognition* algorithm.

Recognition algorithms are used when it is necessary to change the current presentation of the group for the sake of efficiency, or when the problem arises as a basic subcase of the general matrix recognition problem, which can be described as follows. Consider a subgroup $G \subset GL(N,q)$ generated by a set $S$ of matrices over the finite field $GF(q)$. We want to recognize the type of $G$ and answer some basic questions about it. The celebrated Aschbacher's classification of subgroups of $GL(N,q)$ allows one to reduce the problem to a number of base cases, one of which is that of the simple group $SL(n,q)$. Our algorithm will complete the recognition of this case.

Various Monte Carlo *non-constructive* (i.e. producing no explicit isomorphism) algorithms exist for deciding whether a subgroup $G \subset S_n$, or a subgroup $G \subset GL(n,q)$, defined by a generating set $S$, contains respectively $A_n$ or $SL(n,q)$. Our constructive recognition algorithms answer these questions as well, and although they have somewhat larger complexity, their final decision is *Las Vegas*. A Monte Carlo randomized algorithm may give an incorrect answer, with probability less than some $\varepsilon < \frac{1}{2}$, while a *Las Vegas* randomized algorithm never gives an incorrect answer: its output is either a true answer, or a *failure*, which should occur with a fixed probability less than $\frac{1}{2}$ (in which case the algorithm needs to run again).

We develop efficient techniques for computing with transvections in a black box matrix group containing $SL(n,q)$ (or any central extension of $PSL(n,q)$), and for finding transpositions and long cycles in a black box group isomorphic to $S_n$.

## Acknowledgments

I would like to thank my advisors, Egon Schulte from the Department of Mathematics, and Gene Cooperman and Larry Finkelstein from the College of Computer Science. Without their help and encouragement this thesis would never have been completed. I am also grateful to Igor Pak for many enlightening conversations that helped to make many a complicated mathematical issue concrete and approachable.

I am greatly indebted to my Computer Science advisors, Gene Cooperman and Larry Finkelstein. Gene Cooperman has introduced me to the field of Symbolic Computation, and provided me with invaluable guidance through such areas of Computer Science that remain a closed book to many mathematicians. Larry Finkelstein has helped me bridge the traditional gap between pure algebraic results and their algorithmic applications, and, together with Gene Cooperman, has patiently taught me a great number of things necessary for working with finite groups.

I gratefully acknowledge the permissions given by my co–authors Gene Cooperman, Larry Finkelstein and Igor Pak to use in this thesis a number of results obtained in our joint work.

I am indebted to Mikhail Malyutov for his comments on the probabilistic part of the algorithms.

Finally, I thank my wife Anna Shubina for her love and devotion.

Sergey L. Bratus

# Contents

# Chapter 1

# Introduction

## 1-1 Background and Review of Literature

This thesis is motivated by the ongoing search for better algorithms for computation with finite groups. A lot of progress has been made in this direction since the basic methodology for computation with permutation groups was developed by C.C. Sims and others in the 1960s–1970s (see [Si]). We currently possess efficient polynomial-time algorithms for membership testing in finite permutation groups, for checking if a subgroup of a permutation group is normal, for computing the composition series and the size of a permutation group given by a finite set of generators. Permutation actions of finite groups are well understood, and there exist fast (although not provably in polynomial time) heuristics for construction of centralizer and normalizer subgroups (see [CFDi]), as well as for coset and double coset enumeration under such actions (see [Laue]). There exist several computer algebra packages (most notably, GAP and MAGMA, as well as popular general-purpose commercial products like Maple or Mathematica) that implement these algorithms. It is algorithms for computation with matrix groups over finite fields that attract the most attention of researchers nowadays.

Before we go on to discuss the state of affairs in the area of computations with finite matrix groups, let us briefly review some of the facts that make the approach pioneered by Sims [Si] and based on the Schreier algorithm so successful. A full exposition of relevant results and algorithms can be found in [Si].

A permutation group $G$ acting on a finite set $X$ admits a decomposition into a low-index chain of subgroups

$$\{e\} = G_0 \subset G_1 \subset \ldots \subset G_N = G$$

in which the subgroups $G_i$ are stabilizers of an increasing chain of subsets of $X$. The index of each $G_i$ is $G_{i+1}$ is bounded by the permutation degree of $G$, $\deg(G) \leq |X|$. Once the elements of the set $X$ are ordered (they can be identified with integers for simplicity of notation), the stabilizer chain is defined. Given a set $S$ of permutations generating $G$, the Schreier algorithm constructs a larger generating set $S'$ such that each of the groups $G_i$ in the stabilizer chain is generated by a subset of $S'$. $S'$ is called a *strong generating set* of $G$. Knowing $S'$, one can construct a set of *transversal elements* (by taking one representative from each coset $G_{i+1}/G_i$, $i = 0, \ldots, N-1$) such that each element $x \in G$ has a unique representation as a product of these elements.

A pre-computed strong generating set allows one to answer many questions about $G$ very efficiently (and in polynomial time), in particular, test membership in $G$.

It is the low-index property of the stabilizer chain that makes representing a permutation group by its strong generating set and computations involving such a set so efficient. For permutation groups, the complexity of algorithms using strong generating sets naturally depends on the indices of the subgroups in the chain. Since the latter are bounded by the permutation degree of $G$, the complexity of such algorithms "scales well" (i.e. polynomially) as the permutation degree of $G$ grows. Indeed, the Schreier algorithm enhanced by work-saving sifting techniques, makes computations with permutation groups of permutation degree as high as $10^7$ (see [CFTY]) or even $10^8$ (see [We]) still practicable.

Success enjoyed by the methods for permutation groups based on the Schreier algorithm motivated the work of Butler and Cannon, and later by Murray and O'Brien. They focused on analysis of stabilizer chains arising from permutation actions of matrix groups over finite fields. A finite matrix group possesses several permutation representations connected with its natural linear action. For example, it can be viewed as acting on the vectors of a vector space or on subspaces (of some fixed dimension) of a vector space. Unfortunately, the indices in stabilizer chains arising from such actions are bounded only by the number of vectors in the vector space, which depends exponentially on the natural parameters of matrix groups (a permutation action of $G \subset GL(n, q)$ can involve on the order of $q^n$ points). Thus, unlike for permutation groups, the stabilizer chain methods for matrix groups do not scale well. This hardly comes as a surprise, since mathematical intuition tells us that the fundamental difference in the structure of the finite matrix and permutation groups should be reflected in the related algorithmic structures if we want them to be efficient in the general case.

Nevertheless, constructing permutation representations of matrix groups remains an important technique of computing with matrix groups. Other permutation actions of a specific matrix group $G$, especially

- the action of $G$ on an orbit in a vector space $V$ on which $G$ acts linearly,
- the action of $G$ on its cosets by some subgroup $H \subset G$ of a finite index, and
- the action of $G$ on one of its conjugacy classes

can be used for such constructions. The latter were successfully used by Cooperman, Finkelstein, Tselman and York [CFTY] to construct and analyze permutation representations of the Lyons sporadic simple group, and by Weller [We], who constructed a permutation representation of another sporadic simple group $J_4$. These results showed that changing the computational representation of a finite group (maybe several times!) can be quite beneficial for the computation, and can indeed make the difference between a computationally feasible and an infeasible problem. Several important techniques of switching between the presentations of a finite group for the benefit of the computation are discussed in the survey [CF].

The absence of a general technique that could exploit the special structure of finite matrix groups resulted in a very different class of algorithms coming to prominence in this area. Algorithms of this class make a point of not using any representation-specific properties of the group, while heavily relying on randomized methods. In effect, the group is abstracted as a *black box group*, an object that hides any details of group elements' representation or actual identity, giving the user only the raw multiplication table of the group, i.e. providing only ways to compute the product of two elements, the inverse of an element and recognize whether the given element is the identity $e \in G$, in polynomial time. The elements of such an abstract group can be identified, for the purposes of the computation, by arbitrary symbols or bit strings (of bounded length), a query into the multiplication table taking, for example, two such element code names as input and returning just such a code name as output. Any group for which the answers to these questions can be "effectively computed" can be abstracted as a black box group. Saying that an algorithm is *black box* amounts to saying that it does not rely on the particular representation of the group. It turned out that black box algorithms were often not significantly weaker for matrix groups than those relying on the explicit knowledge of the matrix presentation. At the same time, introduction of the black box language and setting greatly helped to clarify the issues of random element generation in matrix (and permutation) groups and to highlight the use of randomization techniques in group algorithms in general.

The term *black box group* was introduced by Babai and Szemerédi in [BaSz]. From the very beginning *black box* methods were closely connected with randomized methods, and used techniques such as random walks on groups. Black box algorithms often rely on our ability to produce random (nearly) uniformly distributed elements of the group by performing a polynomial number of multiplications of its generators. This problem turns out to be a complicated one, with many fascinating aspects.

Babai [Ba1] developed an algorithm to find $\varepsilon$–uniform random elements of $G$ for a fixed $\varepsilon > 0$

with the time requirements of $O(\log^5 |G|)$ for pre-computing the necessary data structures, and then $O(\log |G|)$ for finding each random element. An $\varepsilon$–uniform random distribution for $G$ is one in which the probability of choosing any $g \in G$ lies within $[\frac{1}{|G|} - \varepsilon, \frac{1}{|G|} + \varepsilon]$. Several heuristics exist that are much less demanding than Babai's theoretical bound of $O(\log^5 |G|)$ (see Babai [Ba2] for discussion).

While certainly less efficient for permutation groups, black box algorithms for finite matrix groups often perform no worse than those using the explicit matrix representation, and some of the fastest algorithms developed for matrix groups are almost straight adaptations of their black box versions, using very little information about the actual representation of the elements of the group (e.g. see [CFDi] regarding the algorithms for computing the normal closure of a matrix group).

Black box methods can also be used in the situations when the group $G$ given to us for a computation is presented in such a way that its relation to its natural representation $\Gamma$ is non-transparent. For instance, a $G \cong SL(n,q)$ may be given as a subgroup of $S_N$ for some $N$, generated by several $x_i \in S_N$ (in which case constructing an explicit isomorphism $\Theta : G \to SL(n,q)$ may become necessary for interpretation of results). They may also be useful when the underlying representation of $G$ can no longer be used because the computation has been transferred to a section of the group which can no longer be represented using the same space, or when there is no gain in utilizing the original representation of $G$.

Black box methods became a primary tool in computations with matrix groups over finite fields. Then the need to answer basic questions about subgroups of $GL(N,q)$ generated by a given set of matrices lead to the formation and development of the matrix group recognition project, a general program of effective computation in matrix groups. *Not surprisingly, this program uses many ideas of the black box methods.*

The starting point of the matrix recognition program was the problem, posed by Neubüser, of whether a subgroup of $GL(n,q)$ given by a set of generators, could be recognized to contain or not contain the special linear group $SL(n,q)$ by an efficient algorithm. This question is one of the algorithmic questions suggested by Aschbacher's classification [Asch] of the subgroups of $GL(n,q)$. The solution to the latter recognition problem proposed by Neumann and Praeger in [NeP] motivated others to study similar recognition problems for classical groups. Another solution to the $SL(n,q)$ recognition problem, based directly on Aschbacher's classification, was proposed by Celler and Leedham-Green in [CL1]. A program based on decomposition of matrix groups was developed soon thereafter, leaving the base cases — in particular, the simple groups — as the cases of importance.

This thesis deals with the *black box recognition problem* in two such base cases. We receive a black box group $G$ known to be isomorphic to a classical group $\Gamma$ ($\Gamma \cong S_n$, $\Gamma \cong A_n$, $\Gamma \cong SL(n,q)$ or $\Gamma \cong PSL(n,q)$) and construct an explicit isomorphism $\Theta : G \to \Gamma$, in time polynomial in $n$ in the permutation group setting, and in $n$ and $q$ in the matrix group setting. This problem arises in two ways:

- either in the course of a recognition computation that uses Aschbacher's classification to decompose the computation down to the case of a simple group $SL(n,q)$ ($PSL(n,q)$) or $A_n$ at which point the recognition question for that base case needs to be answered, or
- in the course of a computation when a change of representation has become necessary or expedient.

The same algorithm will also determine the type of the group $G$ if it is known that $SL(n,q) \subset G \subset GL(n,q)$ or that $G$ is isomorphic to a central extension of $PSL(n,q)$. Since the algorithm solves the recognition problems for $SL(n,q)$ and $S_n$ *constructively*, i.e. provides an explicit isomorphism $\Theta : G \to \Gamma$ between the black box and its classical counterpart, its results are Las Vegas, unlike some of the previous *non-constructive* recognition algorithms, which are essentially Monte Carlo by the nature of the randomized methods they employ.

The strategy for handling matrix black box groups that come in an abstract, representationfree form, is a focus point of this thesis. We show how to construct a system of standard generators of $G$ such as transposition or transvections, and how to operate with them in a black box setting and use their properties for establishing the identities of arbitrary elements of $G$ under a isomorphism $\Theta$ fixed by several initial choices of elements in the black box group $G$. Our approach generalizes that of Cooperman, Finkelstein and Linton in [CFL].

For an algebraic symbolic algorithm to be efficient, it must be based on the structural properties of the algebraic objects it manipulates. In particular, matrix group algorithms must rely on the underlying geometry of $GL(n, q)$. This thesis presents such a case, by solving the *black box recognition* problem for $SL(n, q)$ using the underlying geometry of transvections (based on the approach of [CFL]).

Algorithmic considerations often result in exciting mathematical problems, and we hope that the present thesis will not fail to provide another example of this. Algebraic symbolic computation systems such as GAP (Groups, Algorithms, Programming) now provide mathematicians with a wide range of much appreciated (and previously unheard of) opportunities to test non-trivial hypotheses by conducting computational experiments, or, putting it simply, to "quickly try things out". While certainly no substitute for mathematical intuition and understanding of the basic notions, these tools are so efficient exactly because their algorithms incorporate deep structural results about the underlying mathematical objects. Conversely, a deeper understanding of the structure of a mathematical object nowadays quickly translates to elegant algorithms. In other words, mathematics that goes into or uses symbolic computation algorithms is not a different, "brute force" kind of mathematics, as opposed to the "pure" kind.

## 1-2 General group recognition problem

In this section we describe the group recognition problem in full generality and describe the current state of affairs in this area. In later sections we will present solutions to particular recognition problems – those for $S_n$, $A_n$, $PSL(n,q)$ and $SL(n,q)$.

The following loosely stated question can be considered the motivation for all flavors of recognition problems:

Given a classical group $G$, say $G = S_N$ or $G = GL(N,q)$, and the subset $\{x_1, ..., x_n\} \subset G$, describe the basic properties of the subgroup $H = \langle x_1, ..., x_n \rangle \subset G$.

The basic questions posed about $H$ include

1. **ORD** Finding the order of $H$
2. **NORM** Deciding if $H$ is a normal subgroup of $G$
3. **SIMP** Deciding if $H$ is a simple subgroup of $G$
4. **NCREC** Deciding if $H$ contains a subgroup $\Gamma$ isomorphic to $A_N$ for $G = S_N$, or $SL(N,q)$ for $G = GL(N,q)$.
5. **REC** If it is known that $H$ is isomorphic to a known group $\Gamma$ of classical type such as $A_n$ or $SL(n,q)$ $(n \leq N)$, construct an isomorphism between $H$ and $\Gamma$.

Clearly, the last two questions are related, and can be posed and solved in succession in order to recognize $H$ completely. It is also clear that the algorithm that solves the last of these questions will answer many further questions about $H$. Questions of the type **NCREC** are usually referred to as *non-constructive recognition*, while questions like **REC** requiring the construction of an explicit isomorphism are known as *constructive recognition*.

The algorithms that answer questions 1–4 for permutation groups ($G = S_N$, $N$ is known, and the generators $x_i$ of $H$ are given, say, by their cycle forms) are well-known. Neumann and Praeger (see [NeP]) proposed a probabilistic non-constructive algorithm to answer question **NCREC** for $GL(N,q)$ (with known $N$ and $q$). Celler and Leedham-Green presented a constructive algorithm to question **REC** in the situation when $x_i$ are known explicitly as matrices in $GL(N,q)$ and the subgroup $H$ to be recognized is $SL(N,q)$. Cooperman, Finkelstein and Linton presented a constructive algorithm for recognition ( **REC** ) of $H \cong SL(N,2)$ in a much more general case when $G$ was allowed to be a genuine *black box* group, i.e. an abstract group with the group operation performed by an oracle (for rigorous definition of *black box* see Sec. 1-3). In particular, their solution, being independent of the group's presentation, makes it possible to change the context of computations from a permutation action of a group to its linear representation (and would work, for instance, for $H$ given as a subgroup of $S_N$ generated by some permutations).

In this thesis we answer the constructive recognition question **REC** in the black box setting for groups known to be isomorphic to $S_n$, $A_n$, $PSL(n,q)$ and $SL(n,q)$ for $n \geq 3$ (in the latter part we build on ideas of [CFL]). Further, we describe a general approach for computations with black box matrix groups containing $SL(n,q)$ that uses the geometry of transvections in $SL(n,q)$. We also discuss handling and recognition of black box permutation groups. In spite of having a probabilistic part, our constructive recognition algorithms are *Las Vegas*.

At the same time a different algorithm for constructive recognition of black box groups of Chevalley type (including the classical case of $SL(n,q)$), was proposed by Kantor and Seress [KS], based on an earlier approach of Cooperman, Finkelstein and Linton. Although their algorithm allows one to recognize classical groups other than central extensions of $PSL(n,q)$, it has significantly larger complexity (possibly requiring as many as $O(q^3n^3 + q^2n^6 + n^9q)$ black box multiplications), which makes our algorithm preferable for practical implementation for the cases of $G \cong SL(n,q)$ or any perfect central extension of $PSL(n,q)$.

## 1-3 Black box groups and black box recognition algorithms

In this section we give a rigorous definition of the *black box* group. The term was introduced by Babai and Szeremedi (see [BaSz]). The definition in question is designed to capture a wide range of cases when the presentation of the target group $G$ is not natural, and its relation to the classical permutation or linear representation of $G$ is not transparent. Essentially, we have only the raw multiplication table of $G$, realized by an oracle that performs lookups into this table.

**Definition 1-3.1** *A finite group $G$ whose elements are encoded by binary strings of uniform length $M$, and group operations are performed by an oracle (the black box) is called a black box group. The oracle can compute the product of elements, the inverse and recognize the identity element in time polynomial in $M$.*

The time $\mu$ required for the oracle to perform one multiplication is the natural complexity parameter of black box algorithms. The time $\mu$ usually depends on the representation of the black box group and grows as $M$ grows. In practice, the number of multiplications dominates (or is commensurate with) the number of the other two black box operations, so it can be taken as a measure of complexity. An efficient algorithm should require no more than a polynomial number of black box multiplications.

Note that the requirement that the identity element be recognized in time polynomial in $M$ excludes some commonly used group presentations, such as the presentation by generators and relations, from being considered as black box.

Algorithms for computing with black box groups rely more on the internal structure of the group and less on the existence of a concrete matrix or permutation representation. They can be used when it is no longer possible (or profitable) to use the existing representation.

This becomes important in the course of many computations where the underlying representation (in terms of matrices or permutations) can no longer be used because the computation has been transferred to a section of the group which can no longer be represented using the same space. Alternatively, black box algorithms can be useful when there is no gain in utilizing the existing representation. An example of this is the computation of the normal closure of a matrix group in which the fastest known algorithms are straight adaptations of methods developed for black box groups (see [Asch]) and use the existence of the matrix representation only to bound the subgroup chain length.

In recognition problems some claim is made concerning the actual identity of $G$ (say, $G \cong GL(n,q)$). However, no explicit isomorphism between $G$ and its classical representation is given, and the task of a recognition algorithm is to construct such an isomorphism. It is also possible that $G$ is known to belong to a specific classical series (say, $SL(\cdot,\cdot)$), but its parameters ($n$ or $q$ or both) are not known and should be deduced by the algorithm. While in our main examples $n$ for $S_n$, $A_n$ and $n, q$ for $SL(n,q)$, $PSL(n,q)$ are presumed to be known, we also address recognition problems of the latter type.

The binary strings that encode the elements can be thought of as either encoded actual matrices or permutations (which we have no way of decoding), or simply symbolic names of elements of $G$. The constant $M$ is often used to place an upper limit on the order $|G|$ of the black box group: the group may contain no more than $2^M$ different elements. For $G \subset GL(N,q)$ this gives $M \sim N^2 \log q$, which can be used to estimate $N$ or $q$. Such estimations are important for expected complexity of probabilistic parts of recognition algorithms, especially non-constructive ones.

Some further conditions may be imposed to ease the recognition task. For instance we may demand that the elements of $G$ are *uniquely encoded*, that is, the check for equality of two elements is trivial and doesn't affect the algorithm's complexity. This issue is important if we want a canonical form for each $g \in G$, e.g. for using it in a hash table.

It is clear that one of the operations we need to perform very often in the black box setting is finding the order of an element $x \in G$. A naive method is to check all powers of $x$ until the result in the identity $e \in G$; a better method allows to find $|x|$ after about $\log\log|G|$ multiplications, see [CL2]. The task of finding the order of an element may be delegated to another oracle for simplicity. Such groups we will call *gray box* groups; (While slightly different meanings are attributed to the term by various authors, there seems to be no established tradition at this point.) A "gray box" complexity estimate is easy to convert to black box complexity, once the method for finding orders is specified.

A non-constructive recognition algorithm usually arrives at a decision by inspecting a series of random elements of $G$ and performing black box operations on them. The algorithms for **NCREC** typically search for a special kind of element, say $x \in G$ having special cycle forms (which can be deduced from the order $|x|$) in the permutation action setting, or an element acting irreducibly on a vector subspace of given dimension (which can also be deduced from the order, as discussed in the next section). The number of trials is determined either by $|G|$ if known, or else an estimate of the form $|G| \leq 2^M$ where $M$ is the length of bit strings in the black box group computational representation. In case a classical group $\Gamma$ is known, for which $G \cong \Gamma$, the required number of trials — for a fixed bound on the probability of error — can be expressed via parameters $N$ and $q$ (we primarily deal with such favorable situations). If an element of the desired special form is found, the algorithm answers the question affirmatively, otherwise the answer is negative. This procedure is essentially Monte Carlo. A constructive recognition algorithm, typically of much larger complexity, is required to make the answer Las Vegas.

The output of a constructive algorithm is the constructed isomorphism $\Theta : G \to \Gamma$ for a black box group $G$ and a classical (and naturally represented) group $\Gamma$. This output consists of necessary data structures and a method for finding the image $\Theta(x)$ of an arbitrary $x \in G$. In effect, the output is another algorithm which re-uses the same pre-computed data structures.

Both the constructive and the non-constructive algorithms hinge on the availability of an algorithm that produces independent and uniformly distributed random elements of $G$ by means of the black box operations. The complexity of such an algorithm constitutes a large part of the overall complexity of black box algorithms. Further, the analysis of Babai [Ba1] suggests that guaranteeing uniform distribution of the resulting random elements is prohibitively expensive.

However, the distribution can be made $\varepsilon$*–uniform*, i.e. as close to uniform as desired, so that the probability $P(x)$ of any $x \in G$ to be chosen obeys

$$|P(x) - \frac{1}{|x|}| < \varepsilon \qquad \forall x \in G$$

Babai offered an algorithm that finds $\varepsilon$–uniform elements of $G$ after $O(\log |G|)$ group operators each, after a one-time preprocessing phase of $O(\log^5 |G|)$ group operations. Existing implementations use various heuristics for faster random element generation, most commonly the *random subproducts* technique (see [CLMNO]).

The problem of generating random nearly uniformly distributed elements of a black box group is a very important and a truly fascinating one. The current techniques use various random walks on $G$, so that the complexity of the algorithm is determined by the mixing time of the walk, i.e. the number of steps after which the walk can be considered to be at random. For the discussion of the state of the art in random group element generation see Babai [Ba2]. We quote some of this discussion in Sec. 2-2.

## 1-4 Statement of results

In this section we summarize the new results presented in this thesis. Apart from the two theorems below, our contribution includes new methods for computing with transvections in a black box matrix group and for constructing the computational model of the underlying finite field $GF(q)$ given a black box matrix group over $GF(q)$.

**Theorem 1-4.1** *Let $G$ be a black box group specified by a generating set $\mathcal{G}$ such that $G$ is known to be isomorphic to the permutation group $\Gamma$, $\Gamma = S_n$ or $\Gamma = A_n$ for a known $n$. Let $\mu$ be the time required to perform a group operation in $G$, $\rho$ the time required to compute a (nearly) uniform random element of $G$, let $\omega$ be the time required to find the order of an arbitrary element of $G$. Then, provided that Conjecture 1-4.3 below holds, a Las Vegas algorithm exists* *which in time* $O([\mu+\rho+\omega]n\log^2 n)$ *can perform the following operations:*

- *produce a data structure which can then be used to compute an isomorphism* $\Theta : G \to \Gamma$ *for which each invocation* $\Theta(x)$, $x \in G$ *takes time* $O\left(\mu n \log n\right)$.

In the case when $n$ in the previous theorem is not known, but we know an upper bound for it, we prove the following

**Theorem 1-4.2** *In the conditions of the previous theorem, given an upper bound* $N$ *for* $n$*, and provided that Conjecture 1-4.3 holds, a Monte Carlo algorithm exists which in time* $O([\rho+\omega+\mu]N\log^2 N)$ *can find* $n$*, and which can be made Las Vegas by adding a checking phase that verifies, with probability of error less than* $\varepsilon > 0$*, if* $G \cong S_n$ *for a given* $n$*, in time*

$$O\left([\rho+\mu+\omega]\log\left(\frac{1}{\varepsilon}\right) n\,\log^2 n + \mu\, s\, n \log n\right).$$

An upper bound, $N$, can be derived from the uniform binary string length $M$ of black box group by observing that $n! \leq 2^M$.

The algorithm depends on properties of elements of special cycle forms (in particular the long cycle and a transposition) that can be verified by computing the orders of these elements alone. Assuming the following conjecture gives us an elegant and efficient method of locating such elements.

**Conjecture 1-4.3** *For every even* $n \geq 20$*, there exist such primes* $p_1 > p_2 > 3$ *that* $p_1+p_2 = n$*. Moreover, the number* $\pi(n)$ *of such pairs is*

$$\pi(n) > \frac{1}{3}\frac{n}{\log^2 n}$$

This conjecture is immediately recognizable as a slightly extended form of the famous Goldbach Conjecture. The latter, while not yet demonstrated, has been checked for up to astronomically large numbers. We discuss the legitimacy of using this conjecture in Sec 3-3.

For matrix black box groups we offer the following theorem.

**Theorem 1-4.4** *Let $G$ be a black box group specified by a generating set $\mathcal{G}$ with each element of $G$ represented by a bit string of length $N$. There is a Las Vegas algorithm for verifying that $G$ is isomorphic to a central extension of $PSL(n, q)$ for a specified $n$ and $q$. The algorithm has input ($\mathcal{G}$, $n$, $q$) and takes time*

$$O(|\mathcal{G}|N + n^3 \log n (\log q)^4 + nq\rho + n^3\epsilon + n^2\mu q \log q)$$

*where $\mu$ is the time required to perform a group operation in $G$, $\rho$ is the time required to compute a (nearly) uniform random element of $G$, and $\epsilon$ is the time for a field operation in $GF(q)$. If $G$ has the required isomorphism type, then the algorithm will confirm this with probability greater than $1/2$ and produce data structures for deterministically computing:*

- *the order of the center of $G$, and hence the isomorphism type of $G$,*
- *a homomorphism $\Theta : SL(n, q) \longrightarrow G$, in which the time to compute $\Theta(\xi) \in G$ takes time $O(n^3\varepsilon + n^2\mu \log q)$ and the time to find $\xi \in \Theta^{-1}(x)$, $x \in G$ takes time $O(N + n^2 q\mu)$.*

*Otherwise, the algorithm will return failure.*

# Chapter 2

# Background for black box recognition

In this section we discuss the basic technique of handling elements of a black box group $G$. These techniques are based only on the output of the black box oracle (which realizes the multiplication table of $G$ and knows how to check if an element output as a result of a multiplication is the identity) and on our knowledge of the (classical) type of $G$, never on any information about the actual identity of the elements of the black box group, except when deduced from the outputs of the oracle.

## 2-1 Common techniques

The oracle for group multiplication can be used in the following ways:

### 2-1.1 Using the order of $x \in G$

When the structure of the black box group $G$ is known (as it will be in the recognition problems we address in this thesis), the order $|x|$ of an element $x \in G$ can provide important information about the identity of $x$. In this section we discuss various ways of using $|x|$ for recognition purposes once we know it.

Computationally, finding the order of $x \in G$ may require as many as $|x|$ black box multiplications, i.e. the "naive" method of checking all powers of $x$ until one gets the identity cannot be improved upon (for example, as Babai pointed out, if $G$ is a cyclic black box group). In the case of matrix groups computing orders of elements can be made faster, and in many cases certain divisibility properties of $|x|$ can be checked without finding $|x|$ explicitly.

In the setting of our black box group $G \cong S_n$ or $G \cong A_n$ we use order to deduce cycle forms of elements, and to find elements of a special cycle form in a series of random trials. For example, an element $x \in G$ such that $|x| = p_1 p_2$ where $p_1$, $p_2$ are prime and $p_1 + p_2 > n/2$ is easily seen to consist of two cycles of lengths $p_1$ and $p_2$ respectively. We use similar considerations to recognize and construct elements of $G$ which are provably long cycles or transpositions.

In the matrix black box setting the divisors of $x$ can impart information about whether $x$ acts irreducibly in the space of the natural linear representation of $G$ or in a subspace of smaller dimension. It is the order of $SL(n, q)$ together with a number–theoretical result of Zsigmondy that makes such deductions possible. Namely,

$$|SL(n,q)| = q^{\frac{n^2-n}{2}}(q^n - 1)(q^{n-1} - 1)(q^{n-2} - 1)\dots(q^2 - 1)\,,$$

that is, the divisors of $|SL(n,q)|$ (and the orders of its elements) are those of $q^i - 1$ for $i = 2, \dots, n$. An element $x$ whose order is known to contain a divisor of $q^n - 1$ but not of any $q^i - 1$ for $i = 2, \dots, n-1$ is easily seen to act irreducibly under the natural representation of $SL(n, q)$. Similar statements can be made about the action of an element $y$ such that $|y|$ is known to contain a divisor of $q^k - 1$ but not of $q^i - 1$ for $i < k$ under a linear representation of dimension $k$ ($k < n$). A result of Zsigmondy guarantees the existence of such prime divisors for almost all pairs $(q, n)$, which makes these considerations fruitful. For a rigorous treatment see Sec. 4-4 or [NeP]. This technique allows recognition of irreducible or *nearly irreducible* (see [NeP]) elements of $G \cong SL(n,q)$ in a series of random trials, and, taken further, construction and recognition of elements that are provably transvections in $G$. Recall that a transvection in $SL(n,q)$ is defined as an element $\tau \in SL(n,q)$ such that it fixes (pointwise) a hyperplane in $V = (GF(q))^n$, where $V$ is the vector space of the natural matrix representation. It follows that a transvection is completely defined by the hyperplane it fixes (called the *axis* of $\tau$), and by giving a single vector in that hyperplane (this vector is referred to as the *center* of $\tau$). The properties of transvections which are important for our recognition purposes will be discussed in detail in 4-2. An alternative description of transvections in $SL(n,q)$ for $n > 3$ is that they form the conjugacy class of the elementary matrices, i.e matrices $I + E_{ij}$, $i \neq j$, where $I$ is the identity matrix and $E_{ij}$ is a matrix with only one non-zero entry at $(i,j)$.

### 2-1.2 Obtaining more elements with a known property through black box multiplication

After we gained enough information from using the oracle (say, by locating, after a series of random trials, an element $x$ whose order $|x|$ guarantees some special property), we can "propagate" that property through $G$. That is, we can now use the oracle to obtain more elements of $G$ with that property. For example, conjugation by random elements of $G$ (produced by the "uniform random element" algorithm) will produce more elements in the conjugacy class of $x$, i.e. if $x$ were known to be a transposition in $G \cong S_n$ or a transvection in $G \cong SL(n,q)$, we can get more (nearly) uniformly distributed transpositions or transvections, as many as necessary.

Conjugation is not the only example of a situation when we use the oracle to obtain an element $z \in G$ with special properties by feeding in two elements $x$, $y$ about which some facts are already established. In fact, this technique is the key to our use of the intersection geometry of transvections. Namely, the oracle and random elements of $G$ can be used to obtain transvections that stabilize the same hyperplane but have different centers, or share the same center while stabilizing different hyperplanes.

In the symmetric group setting (consider $G \cong S_{2m}$ for the sake of this example) the oracle is used to produce a transposition after an element $x$ of the a cycle form $(p_1, p_2, 2)$ is located in a series of random trials in $G$, where $p_1 > p_2 > 2$ are primes and $p_1 + p_2 = n - 2$. The transposition is produced by letting the oracle compute $x^{p_1 p_2}$, whereby the unwanted prime cycles are "multiplied away".

Clearly, this trick of multiplying away the unwanted cycle components works for any cycle form with mutually prime lengths of the cycles, and is used repeatedly because of its simplicity and independence of anything else other than orders of group elements. This example is also quite typical for the matrix black box setting, where the same idea is used to "multiply away" the Singer cycle components of the matrix black box group's elements (see [NeP]) to get an initial (arbitrary) transposition.

### 2-1.3 Using commutation relations for recognizing an element by its action

Much information about elements of a black box group can be recovered by checking commutation relations between them. While this certainly is a trivial observation, it is interesting that actually *all* information (up to an isomorphism) about the elements of a black box group isomorphic to $S_n$ or $SL(n, q)$ (and their simple counterparts $A_n$ and $PSL(n, q)$) can be recovered that way.

We will consider transpositions in $S_n$ and transvections in $SL(n, q)$ (and their analogs in $PSL(n, q)$) as basic elements for our recognition algorithms. Commutation relations for these are simple and cost very little to check in a black box setting.

For transpositions $s, t \in S_n$ the following well-known alternative depending on the mutual "location" of $s$, $t$ exists:

$$s \cdot t = t \cdot s \qquad \text{or} \qquad s \cdot t \cdot s = t \cdot s \cdot t$$

where the first alternative takes place when either $s$ and $t$ permute four different points of the underlying set or $s = t$. Similarly, for a cycle $C$ of length $m > 2$ and a transposition $t$, denoting $s = C \cdot t \cdot C^{-1}$ :

$$C \cdot t = t \cdot C \quad \Leftrightarrow \quad s = t$$

when the sets of points moved by $C$ and $t$ don't intersect, or else

$$(s \neq t,\, s \cdot t \cdot s = t \cdot s \cdot t) \quad \text{or} \quad (s \neq t,\, s \cdot t = t \cdot s)$$

where the second case takes place when only one but not the other of the points transposed by $t$ is moved by the cycle $C$, and the third case takes place when both points of $t$ are moved by $C$.

For transvections $\tau_1$, $\tau_2$ in $SL(n, q)$ similar relations between $\tau_1$, $\tau_2$, $\tau_1 \cdot \tau_2 \cdot \tau_1^{-1}$ and $\tau_2 \cdot \tau_1 \cdot \tau_2^{-1}$ allow us to determine how the centers and the axes of these two transvections are situated with respect to each other, that is, whether the axes of these two transvections coincide (are the same hyperplane), and, if this is not the case, whether the center of $\tau_1$ lies inside the axis of $\tau_2$, and so on.

When applied in the black box setting of $G \cong SL(n, q)$ or $G \cong PSL(n, q)$ to some $t_1, t_2 \in G$ known to be transvections (see the "propagation" technique above), those commutation relations reveal the relative positions of the centers and the axes of the (otherwise mysterious) transvections $t_1$ and $t_2$. This ability to check the "intersection geometry" of transvections in the black box group turns out to be enough for us to be able to write down equations over $GF(q)$ for the coordinates of the center of an arbitrary transvection in the basis formed by the centers of $n$ fixed transvections. The detailed description of this technique, developed in the present thesis, is found in Sections 4-7.3 and 4-6.

The combination of these techniques suggests the following strategy for recognition of black box groups known to be isomorphic to $S_n$ or $SL(n, q)$ (as well as the corresponding simple $A_n$ and $PSL(n, q)$).

## 2-2 Generating random elements of a black box group

Since the only means of producing a new element of a black box group is the oracle that performs the group operations, all methods currently used for this task try to "get random elements" by multiplying the given generators of the group one way or another, and adding the newly computed elements thus obtained to the existing pool for subsequent multiplications. Such algorithms effectively perform a random walk on the group, and the proof of correctness of an algorithm (or the reasoning to support a heuristic) is based on how fast the particular walk *mixes*.

The following description of Babai's method for random element generation that *provably* provides random elements with $\varepsilon$–uniform distribution is taken from [Ba2]. We quote it here due to the central role played by the procedure for random generation in our recognition algorithms.

*The preprocessing proceeds in* $O(\log|G|)$ *phases. In each phase, a random walk of* random length *between* 1 *and* $O(\log^4|G|)$ *is performed on the Cayley graph corresponding to the current generators. The element found when the walk stops is added to the generators. This is repeated for* $O(\log|G|)$ *rounds. The final walk is repeated* $O(\log|G|)$ *times, and the list* $S$ *of* $O(\log|G|)$ *elements obtained forms the input of the construction phase. The random elements to be constructed are* random subproducts *of* $S$*, i.e. elements of the form* $g_1^{\epsilon_1}\cdots g_m^{\epsilon_m}$*, where* $S=\{g_1,\dots,g_m\}$*, and* $\epsilon_i\in\{0,1\}$*; the* $\epsilon_i$ *are chosen by independent unbiased coin flips.*

In practice, the cost of $O(\log^5|G|)$ black box multiplications appears to be excessive. Existing algorithms use heuristics for faster generation of random elements that are purportedly uniformly distributed. The most notable heuristic is that of *product replacement*, described in [CLMNO].

The random walk it uses is one on sets of generating $k$–tuples of $G$, and is defined as follows. One starts with the $k$–tuple of given generators $g_1,\dots,g_k$ of $G$. Each step of the walk consists of randomly choosing two positions $i$ and $j$ in the $k$–tuple and replacing $g_i$ by either

$g_ig_j$ or $g_jg_i$, chosen randomly (other modifications use $g_ig_j^{-1}$ etc.) After a number of steps, the algorithm outputs a randomly chosen element of the $k$–tuple. In spite of this algorithm's seeming simplicity, no satisfactory estimate on the required number of steps is yet available. For further discussion see [Ba2].

An algorithm for producing a random uniform element of our black box group $G$ plays a crucial role in what follows. We assume henceforth that such an algorithm is available. In our complexity estimates we isolate the complexity of producing a (nearly) uniform random element of $G$, denote it by $\rho$ and imply that $\rho$, which essentially counts the number of black box multiplications in the random walk, is proportional to $\mu$, the common factor of our complexities.

## 2-3 Outline of the recognition strategy

Consider the action of $G$ by conjugation on the smallest conjugacy class $\mathcal{C}$ that generates $G$, where $\mathcal{C} = \{$ transpositions $\}$ for permutation groups and $\mathcal{C} = \{$ transvections $\}$ for matrix groups. For any $x \in G$ we construct the image $\Theta(x) \in \Gamma$ under the desired isomorphism $\Theta : G \to \Gamma$ by analyzing the action of $x$ on $\mathcal{C}$ by conjugation.

- Use the "uniform random element" algorithm to produce a series of random elements of $G$. Using the properties deducible from the orders of these random elements, construct an element $t \in G$ such that provably $t \in \mathcal{C}$.

  Start constructing $\Theta$ by fixing the image of $\Theta(t)$ in the corresponding conjugacy class in the classical group $\Gamma$. This choice is an arbitrary one.

- Using conjugation by random elements of $G$, propagate $t$ in $G$ to obtain a set $B = \{\beta_i : i \in \mathcal{I}\} \subset G$, $G = \langle B \rangle$, with special commutation relations between its elements (here $\mathcal{I}$ is the index set).

- To recognize an arbitrary $x \in G$, compute $x\beta_ix^{-1}$ for all $i \in \mathcal{I}$ and use commutation relations between $x\beta_j x^{-1}$ and $\beta_i \in B$ to recognize the elements $x\beta_j x^{-1}$. From this action of $x$ on $B$ recover $\Theta(x)$.

Specifically, for $G \cong S_n$ :

1. Use a series of random elements of $G$ to produce $a, b \in G$ such that $a$ is a transposition and $b$ a long cycle.

   Fix $\Theta$ by saying $\Theta(a) = (1\ 2)$, $\Theta(b) = (1\ 2\ \ldots\ n)$.

2. Construct the system $B$ of transpositions of neighboring points.

3. For any $x \in G$ use its action on $B$ to recognize $x$. For this step we need to compute a system of non-intersecting cycles of different lengths, for use in commutations relations.

For $G \cong PSL(n,q)$ the strategy is similar, but the role of transpositions is played by transvections, a transvection $t \in G$ is produced from elements that emerge in a series of random trials, and $\Theta(t)$ is fixed arbitrarily. Elements $x$ of $G$ are recognized by their action on the system $B$ of $2n-1$ transvections, and additional data structures are computed to help in commutation relations. Recognition of $G \cong SL(n,q)$ requires an additional step, because the latter group has a center that acts trivially under the conjugation action of $G$ on itself, and hence cannot be recovered from it. We take the easy route and construct, element-wise, the entire center of $G$, which is not large anyway.

This strategy will work as long as we have a way of locating, in reasonable time, some elements of $G$ that will allow us to use the multiplication oracle to manufacture a $t \in \mathcal{C}$, and as long as we have enough elements in $B \subset \mathcal{C}$ to recover $x$ from its action on $B$. For the recognition of $G \cong S_n$ we use the set $B$ of $n-1$ transpositions, and for the recognition of $G \cong SL(n,q)$ we use the set of $2n-1$ transvections for $G \cong SL(n,q)$.

# Chapter 3

# Recognition of a black box group isomorphic to $S_n$

## 3-1 Preliminaries

In this section we present a constructive algorithm for recognition of black box groups known to be isomorphic to $S_n$ and $A_n$. We offer a simple and fast method for construction of an isomorphism $\Theta : G \to S_n$ (or $\Theta : G \to A_n$), based on the relation between the order of an element in $G$ and its cycle form and a conjecture about relative abundance of elements of a special cycle form (e.g. having exactly two cycles of odd prime lengths $p_1$, $p_2$ such that $p_1 + p_2 = n$ for $n = 2k$) in $S_n$. The latter conjecture is easily recognized to be a slightly stronger form of the famous Goldbach conjecture. Although a result that doesn't depend on the Goldbach conjecture (at the cost of increasing the complexity by an order of $n$) can also be formulated, we choose to base our presentation on the conjecture which has already been verified up to much larger numbers than any reasonable parameters to our algorithm. In the subsequent sections we inform the reader about the current state of affairs with the Goldbach conjecture.

We assume no knowledge of the details of the particular representation of $G$, working directly in the black box setting. When estimating the complexity of the algorithm, we diverge from the traditional black box setting by isolating the complexity of finding the order of an element of our black box group into a separate task. This approach to estimating complexity is known as the *gray box* approach. We will henceforth use the term *gray box* group to mean a black box group $G$ with an additional oracle that, in time $\omega$, computes the order of an element of $G$. The use of this terminology helps distinguish between the complexity inherent to the constructive recognition procedures versus the complexity of auxiliary tasks. Knowledge about the specific presentation of $G$ may be used for these tasks, in particular, for a group realized by invertible matrices over a finite field there exist heuristic algorithms for determining the order of a matrix(see [CL2]). Originally, the term *gray box* group was introduced by F. Celler and C.R. Leedham-Green. We use it in a sense somewhat different from the original meaning.

For the case when no efficient method of finding the order of an arbitrary element of $G$ is available, we give the standard black box complexity, based on an algorithmic shortcut. This shortcut obtains the required information about the orders of elements (essentially, it verifies the aformentioned special cycle form) without actually finding the orders of elements explicitly. The complexity of such a check will take the place of $\omega$ in the complexity estimate for the algorithm.

Another parameter $\rho$ for the complexity of the algorithm is the complexity of the procedure for generating (nearly) uniformly distributed random elements of $G$. A theoretical algorithm for finding random group elements in black box groups in time $O(\log^5 |G|)$ is found in the work of Babai [Ba1], while faster heuristics that appear to work well in practice exist in the work of Celler et al. [CLMNO]. The choice of a particular procedure for random generation is a matter of computational practice; therefore, it is desirable that the inherent complexity of recognition be separated from the complexity of this task in the total estimate.

The idea of our two-phase constructive recognition algorithm for a black box group $G$ isomorphic to $S_n$ is as follows:

In the first phase, we find two elements $a, b \in G$ such that there exists an isomorphism $\Theta : G \to S_n$ that maps $a$ into the transposition $(1, 2)$ and $b$ into the long cycle $(1, 2, \ldots, n)$. Since a transposition and a long cycle generate the entire $S_n$, the choice of $a, b$ determines $\Theta$ completely.

In the second phase, we pre-compute a set $T \subset G$ of elements that map under $\Theta$ to elements of special cycle forms (in particular, $\Theta(T)$ includes $n$ transpositions of neighboring points that generate $S_n$), and then explicitly construct the image $\Theta(x) \in S_n$ for an arbitrary $x \in G$, using the action of $x$ on $T$.

For the case when $G \cong A_n$ the role of the special conjugacy class is played by 3-cycles.

We prove the following theorems:

**Theorem 3-1.1** *Let $G$ be a black box group specified by a generating set $\mathcal{G}$ such that $G$ is known to be isomorphic to the permutation group $\Gamma$, $\Gamma = S_n$ or $\Gamma = A_n$ for a known $n$. Let $\mu$ be the time required to perform a group operation in $G$, $\rho$ the time required to compute a (nearly) uniform random element of $G$, let $\omega$ be the time required to find the order of an arbitrary element of $G$. Then, provided that Conjecture 3-1.3 below holds, a Las Vegas algorithm exists which in time $O([\mu + \rho + \omega] n \log^2 n)$ can perform the following operations:*

- *produce a data structure which can then be used to compute an isomorphism $\Theta : G \to \Gamma$ for which each invocation $\Theta(x)$, $x \in G$ takes time $O(\mu n \log n)$.*

In case when no efficient algorithm for finding orders of elements of $G$ is available, we can extract the necessary information about the orders of elements without finding the order explicitly. In that case in the above algorithm $\omega = O(\mu n / \log n)$, and the overall black box complexity becomes $O(\mu n^2 \log n + \rho n \log^2 n)$.

We also give weaker bounds that do not assume the Goldbach Conjecture. For numerical experiments related to the extended conjecture see section 3-3.

In the case when $n$ in the previous theorem is not known but we know an upper bound for it, we prove the following

**Theorem 3-1.2** *In the conditions of the previous theorem, given an upper bound $N$ for $n$, and provided that Conjecture $2-1.3$ holds, a Monte-Carlo algorithm exists which in time $O([\rho+\omega+\mu]N\log^2 N)$ can find $n$, and which can be made Las Vegas requiring quadratic time in $N$.*

An upper bound $N$ can be derived from the uniform binary string length $M$ of black box group by observing that $n! \le 2^M$. The same value of $\omega$ applies when no efficient algorithm for order computation is available.

The complexity estimates are based on the following conjecture.

**Conjecture 3-1.3** *For every even $n \ge 20$, there exist such primes $p_1 > p_2 > 3$ that $p_1+p_2 = n$. Moreover, the number $\pi(n)$ of such pairs is*

$$\pi(n) > \frac{1}{3}\frac{n}{\log^2 n}$$

The algorithm proceeds as follows. We pick elements $g \in G$ randomly until we find an element of order $|g| = p_1p_2$, where $p_1, p_2$ are primes such that either $p_1+p_2 = n$ (for $n$ even), or $p_1+p_2 = n-1$ (for $n$ odd). The existence of such elements follows from the Goldbach Conjecture. It is clear that such an element consists of exactly two cycles of lengths $p_1$ and $p_2$. We refer to them as *Goldbach elements.* It turns out that there are sufficiently many of them. Next we find one or two (depending on the parity of $n$) transpositions such that together with our Goldbach element they generate the entire symmetric group $S_n$. Finding these transpositions involves a modification of Goldbach elements and some technical details.

Conjugating the transposition by the Goldbach element, we obtain the desired $a, b \in G$ that define the isomorphism $\Theta$. We show that, on average, for the general $n$, our algorithm requires Monte Carlo time $O([\rho + \mu + \omega]n \log^2 n)$. This result is based on existing theoretical results concerning the Goldbach Conjecture and is confirmed by experimental evidence obtained on computers.

Finding a deterministic upper bound in the worst case would require an analysis even stronger than the Goldbach Conjecture itself, and is therefore beyond the scope of this thesis. Instead, we offer a number of simple heuristics and an easy quadratic deterministic upper bound which follows directly from the Goldbach Conjecture.

We state the algorithm first, then prove its correctness assuming the conjecture, and finally elaborate on possible improvements (including possibly getting rid of the conjecture altogether!). We need the following trivial observation: given a transposition $\tau = (i_1, i_2)$ and a permutation $\sigma$, we know that $\sigma^{-1}\tau\sigma$ is also a transposition, and

$$\sigma^{-1}\tau\sigma = (\sigma(i_1), \sigma(i_2)).$$

Therefore if $\sigma$ is (nearly) uniformly distributed in $S_n$, then so is $\sigma^{-1}\tau\sigma$ in the set of transpositions of $S_n$.

We say that a permutation $\sigma$ *touches* a point if it does not fix that point (i.e. the point does belong to the support of $\sigma$). We say that the transposition $\tau = (i_{1,}i_2)$ *touches* the cycle $\sigma = (j_1, j_2, \ldots, j_{p_2})$ if either $i_1$ or $i_2$ or both belong to the support of the cycle $\sigma$.

## 3-2 The algorithm

Our algorithm calls on the procedure for finding the order $|x|$ of an element $x \in G$, and the procedure that produces (nearly) randomly distributed elements of $G$. All of them take as input the set of generators $\mathcal{G} = \{g_1, g_2, \ldots, g_m\}$ of $G \cong S_n$ as a black box group, and the number

$n$, which is the degree of $G \cong S_n$. It outputs two elements $a, b \in G$ such that there exists an isomorphism $\Theta : G \to S_n$ carrying $a$ to $(1,2)$ and $b$ to $(1, 2, \ldots, n)$. Its complexity for the general $n$ is $O([\rho + \mu + \omega] n \log^2 n)$.

Elements $a, b$ output by the algorithm can be input to the procedure that takes an element $x \in G$ (and elements $a, b$ as above) and outputs the permutation $\Theta(x)$ into which $x$ is carried by $\Theta : G \to S_n$ for which $\Theta(a) = (1,2)$, $\Theta(b) = (1, \ldots, n)$.

The algorithms for $S_{2k}$ and $S_{2k+1}$ differ slightly. Namely, when $n$ is odd, we find two transpositions, whereas for even $n$ finding one is sufficient.

### 3-2.1 The algorithm for $S_{2k}$

1. *Locate a Goldbach element in $G$.*

   Find an element $y \in G$ whose order, $|y|$, satisfies

   $$|y| = p_1 p_2,$$

   where $p_1, p_2$ are primes such that $p_1 + p_2 = n$. By the Goldbach Conjecture there will be at least one such pair of primes $(p_1, p_2)$. We will show in section 3-3 that a random search should succeed after Monte Carlo $O(n \log^2 n)$ tries. Notice that the solution of the Goldbach equation $n = p_1 + p_2$ is, on average, far from unique (a computation described in section 3-3 shows that the number of such prime pairs is at least $\frac{1}{3} n / \log^2 n$ for $20 \le n \le 10^6$). An element $y$ as above consists of two cycles of lengths $p_1, p_2$ respectively. Clearly, $y_1 = y^{p_2}$ and $y_2 = y^{p_1}$ are cycles of lengths $p_1$ and $p_2$ respectively. Denote

   $$y_1 = (i_1, i_2, \ldots, i_{p_1}),$$

   $$y_2 = (j_1, j_2, \ldots, j_{p_2}).$$

We can re-number the points so that $\{i_1, i_2, \ldots, i_{p_1}\} = \{1, \ldots, p_1\}$ and $\{j_1, j_2, \ldots, j_{p_2}\} = \{p_1 + 1, \ldots, n\}$.

2. *Locate a transposition in $G$.*

   To this end, search for an element $x \in G$ such that

$$|x| = 2\, q_1\, q_2,$$

   where $q_1 + q_2 = n - 2$, $q_1, q_2$ are odd primes.

   Then $x$ consists of 2 odd prime cycles of lengths $q_1$ and $q_2$ and a transposition. Hence

$$a = x^{q_1 q_2}$$

   is a transposition. Once we have found some transposition, we can easily find any number of (nearly) uniformly distributed transpositions, by conjugating the original transposition by random elements of $G$.

3. *Check if the transposition $a$ touches both $y_1$ and $y_2$. If not, produce and try further random transpositions until we have the desired property.*

   We are looking for a transposition that permutes a point from the cycle $y_1$ with a point from the cycle $y_2$ (i.e. touches both cycles). This is the case if and only if $a$ commutes with neither $y_1$ nor $y_2$. Consequently, we check that

$$ay_1 \neq y_1 a \quad \text{and} \quad ay_2 \neq y_2 a.$$

4. *Compute $b = y_1 a y_2$. This is a long cycle in $G$.*

   If $a = (l_1, l_2)$ commutes with neither cycle, then one of $l_1$ and $l_2$ lies in the support of $y_1$, and the other in the support of $y_2$. Assume, without loss of generality, that

$$\{l_1, l_2\} = \{p_1, p_1 + 1\}.$$

It is easy to see that

$$b = y_1 a y_2$$

is a cycle in of length $p_1 + p_2 = n$ in $G$, and that there is a homomorphism $\Theta : G \to S_n$ that carries $b$ into the long cycle $(1, 2, \ldots, n)$ and $a$ into $(p_1, p_1 + 1)$.

5. *Conjugate $a$ by $b^{p_1 - 1}$ to get $a'$.*

   Now

   $$\Theta(b) = (1, 2, \ldots n), \quad \Theta(a) = (p_1, p_1 + 1), \quad \Theta(a') = (1, 2),$$

   and $a', b$ is the standard pair of generators for $S_n$. As $\{a, b\}$ generate $S_n$ (any transposition together with any cycle of length $n$ do so), we have fixed the isomorphism $\Theta : G \to S_n$, and, in particular, can compute the image of any element $z \in G$ under the desired isomorphism $\Theta$ explicitly (see section 3-4).

### 3-2.2 The algorithm for $S_{2k+1}$

1. *Locate a Goldbach element in $G$.*

   Find an element $y \in G$ such that

   $$|y| = p_1 p_2,$$

   where $p_1 + p_2 = n - 1$. Then $y$ consists of two cycles of lengths $p_1$ and $p_2$ and a fixed point $m$. Heuristically, the probability of finding such an element is high enough, so that after Monte Carlo $O(n \log^2 n)$ tries random search should succeed. We get two cycles $y_1 = y^{p_2}$ and $y_2 = y^{p_1}$ of lengths $p_1$ and $p_2$ respectively. Denote, as above,

   $$y_1 = (i_1, i_2, \ldots, i_{p_1}),$$

   $$y_2 = (j_1, j_2, \ldots, j_{p_2}),$$

   and re-number the points so that

   $$\{i_1, i_2, \ldots, i_{p_1}\} = \{1, \ldots, p_1\} \quad \text{and} \quad \{j_1, j_2, \ldots, j_{p_2}\} = \{p_1 + 1, \ldots, n - 1\}.$$

2. *Locate a transposition in $G$.*

   Look for an element $x \in G$ such that

   $$|x| = 6\, q_1\, q_2,$$

   where $q_1 + q_2 + 5 = n$, and $q_1$, $q_2$ are odd primes $> 3$. The existence of such a pair of primes constitutes a slight extension of the Goldbach Conjecture. The element $x$ will have the cycle structure of $(q_1, q_2, 3, 2)$. The element $a_1 = x^{3q_1q_2}$ is then a transposition. Any number of (nearly) uniformly distributed random transposition can now be obtained by conjugating $a_1$ by random elements of $G$.

3. *Check if the transposition $a_1$ touches both cycles $y_1$ and $y_2$. If not, keep producing and trying random transpositions until successful.*

   The transposition $a_1$ touches the cycle $y_1$ if and only if they don't commute, and similarly for cycle $y_2$ . So, as above, we only need to check that

   $$a_1 y_1 \neq y_1 a_1 \quad \text{and} \quad a_1 y_2 \neq y_2 a_1.$$

4. *Locate a transposition that touches both the fixed point $m$ and one of the cycles ( $y_1$ or $y_2$). Keep producing and trying random transpositions until successful.*

   Obtain a random transposition $a_2$ (by conjugating $a_1$ with a random element of $G$). First check whether $a_2$ commutes with $y_1$ and $y_2$. If $a_2$ commutes with both, or with neither of them, discard the current $a_2$ and obtain a new one. Suppose $a_2$ commutes with $y_2$ but doesn't commute with $y_1$. We need to verify if $a_2$ touches only one point from the support of $y_1$. To this end, check if $a_2$ commutes with $y_1 a_2 y_1^{-1}$ and $y_1^2 a_2 y_1^{-2}$. If $a_2$ commutes with at least one of these, then $a_2$ only permutes the vertices inside the support of $y_1$, and hence it doesn't touch $m$ (see Lemmas 1, 2 of Section 3-3). Special care needs to be taken in the case when $r(y_1) = 3$, which, fortunately, can be easily detected. If the support of

$a_2$ lies entirely inside that of $y_1$, discard $a_2$ and start over. The case when $a_2$ commutes with $y_1$ but doesn't commute with $y_2$ is analogous.

5. *Compute* $b = y_1 a_1 y_2 a_2$ . *This is a long cycle in* $G$

   Indeed, re-numbering the points in a suitable fashion, we can have $a_1 = (p_1, p_1 + 1)$ and $a_2 = (n-1, n)$. Then

$$b = y_1 a_1 y_2 a_2$$

   is a long cycle, carried to $(1, 2, \ldots, n)$ by some isomorphism $\Theta : G \to S_n$.

6. *Conjugate* $a_1$ *by* $b^{p_1 - 1}$ *to get* $a'$.

   We now have the transposition $a'$ and the long cycle $b$ which together generate the entire $G \cong S_n$ and

$$\Theta(a') = (1, 2),$$

$$\Theta(b) = (1, 2, \ldots, n).$$

### 3-2.3 The shortcut to avoid computation of orders

In both versions of the above algorithm we use the order $|x|$ of a random element $x \in G$ for the sole purpose of verifying if $|x| = p_1 p_2$, where $p_1 + p_2 = n$ for $n = 2k$, or $p_1 + p_2 = n - 1$ for $n = 2k + 1$, and verifying two similar properties. All these properties revolve around the Goldbach pairs of primes $p_1$, $p_2$ for $n$ and $n - 1$ ($n$ even), or $n - 1$ and $n - 5$ ($n$ odd).

If the Goldbach pairs $p_1$, $p_2$ are precomputed for these integers, these properties of $x \in G$ can be tested by computing $x^{p_1 p_2}$ and testing it for being the identity, together with its component cycles $x^{p_1}$ and $x^{p_2}$. Each such test involves $O(\log(p_1 p_2))$ multiplications for computing the power and a black box identity test (which is assumed to take time comparable with a multiplication), i.e. no more than $O(\mu \log n)$ complexity per pair. As the conjecture suggests, around $\frac{1}{3} n / \log^2 n$ Goldbach pairs are available in each case. Therefore we need $O(\mu n / \log n)$

complexity for testing each random element for the Goldbach property. This estimate takes the place of $\omega$ in the algorithm's overall complexity when no efficient (i.e. with $\omega$ better that $O(\mu n/\log n)$) trick for order computation is available.

## 3-3 Why this works?

Let us state the extended Goldbach conjecture we are using.

**Conjecture.** For every even $n \geq 20$, there exist such primes $p_1 > p_2 > 3$ that $p_1 + p_2 = n$. Moreover, the number $\pi(n)$ of such pairs is

$$\pi(n) > \frac{1}{3}\frac{n}{\log^2 n}$$

We have verified this conjecture for $20 \leq n \leq 10^6$. It was proven by Hardy and Littlewood (see above) to hold asymptotically for 'almost all' $n$. We will explain the meaning of 'almost all' in the section 3-5.

The first part is a slight extension of the classical Goldbach conjecture as stated by Goldbach in his letter to Euler (June 7, 1742). Conjectures similar to the second part were made by Sylvester in 1871 (see [Sy]), and subsequently refined by others, including Hardy and Littlewood ([HL], see also [W]), who found the following asymptotic formula for $\pi(n)$:

$$\pi(n) \sim 2C\frac{n}{log(n)^2}\prod_{p|n}\frac{p-1}{p-2},$$

where the product is over the odd prime divisors p of n, and

$$C = \prod_{j=2}^{\infty}\left(1 - \frac{1}{j^2}\right).$$

We will need the above conjecture for estimating the probability of finding a Goldbach element.

### 3-3.1 Finding a Goldbach element.

The first step of the algorithm consists of finding a Goldbach element. The Goldbach Conjecture guarantees that such elements exist for all $n$. Let us estimate the probability that a randomly chosen element of $G$ is a Goldbach element. Goldbach elements of order $p_1\,p_2$ form a conjugacy class corresponding to partitions $(p_1, p_2)$ (for even $n$) and $(p_1, p_2, 1)$ (for odd $n$). It is well known that the probability of getting an element $g \in G$ from the conjugacy class corresponding to the partition $(\lambda_1, \lambda_2, \ldots)$ is equal to

$$\frac{1}{\lambda_1\,\lambda_2 \ldots}.$$

Hence the probability of finding an element of order $p_1\,p_2$ in $G$ is

$$\frac{1}{p_1\,p_2} \geq \frac{4}{n^2}.$$

We need to estimate the number of pairs of primes $(p_1, p_2)$ such that $p_1 + p_2 = n$. The following heuristic demonstrates where the factor $n/\log(n)^2$ in the Hardy-Littlewood formula comes from. There are (asymptotically) $\frac{n}{\log n}$ primes less than $n$, hence $\left(\frac{n}{\log n}\right)^2$ pairs of primes. If the sum of such pairs were uniformly distributed, then each number under $2n$ would have an equal chance of being the sum of some $(p_1, p_2)$ . Hence we would have

$$\left(\frac{n}{\log n}\right)^2 / \, 2n = \frac{n}{2\log^2 n}$$

pairs $(p_1, p_2)$ such that $p_1 + p_2 = n$. Although the distribution of sums of pairs of prime numbers is clearly non-uniform, numerical experiments and asymptotic results mentioned above support the conjecture.

Therefore it takes, on average, $O(n \log^2 n)$ steps to get a Goldbach element. In the worst case, it should take at most $O(n^2)$ steps to get a Goldbach element, assuming that there exists at least one pair of primes as stated in the Goldbach Conjecture.

### 3-3.2 Finding an arbitrary transposition.

For subsequent steps of the algorithm we need to find random transpositions. Once a single transposition is found, this can be easily achieved by conjugating it with random elements of $G$ (we assume availability of an efficient algorithm for producing (nearly) uniformly distributed random elements of $G$ ). The question, then, is how to find the initial transposition in the black box setting.

We start with elements in the conjugacy classes corresponding to partitions $(q_1, q_2, 2)$ (even $n$) and $(q_1, q_2, 3, 2)$ (odd $n$). The latter is the only possible cycle form of an element $x \in G$ of order $|x| = 6q_1q_2$, where $q_1 + q_2 + 5 = n$.

The probabilities of getting them are $\frac{1}{2\,q_1\,q_2}$ and $\frac{1}{6\,q_1\,q_2}$ respectively. Therefore the expected number of tries before locating such an element $x$ for fixed $q_1, q_2$ is $2\,q_1\,q_2 \leq 2n^2$ (or $6q_1q_2 \leq 6n^2$ respectively). We expect the number of such pairs $q_1, q_2$ to be of the order $O(n/\log^2 n)$, so the expected time is again Monte Carlo $O(n \log^2 n)$ for general $n$.

### 3-3.3 Transpositions touching cycles.

In both the even and the odd cases we find a transposition that touches each of the two cycles $y_1$ , $y_2$ of prime lengths $p_1$ and $p_2$ ( $p_1 + p_2 = n$ or $p_1 + p_2 = n - 1$). In the odd case we also need a transposition that touches the point $m$ and one of these cycles (i.e. simply permutes $m$ with a point from the support of the cycle).

Notice that, as we have remarked in the Introduction, the permutations resulting from conjugating the initial permutation with (nearly) uniformly distributed elements of $G$ are themselves (nearly) uniformly distributed. The probability that a randomly chosen transposition touches both $y_1$ and $y_2$ is, of course,

$$\frac{p_1 \cdot p_2}{\binom{n}{2}} \geq \frac{1}{n}.$$

Thus it will take no more than $n$ tries to locate the required transposition; much fewer if $p_1$

and $p_2$ are big enough. For instance, should we decide to consider only Goldbach elements with cycles of lengths $p_1 > p_2 > n/6$ , the number of transpositions touching both cycles will be at least $(n-1)/6 \cdot 5(n-1)/6$. Thus the probability that we find such a transposition will be simply constant,

$$\frac{\frac{n-1}{6} \cdot \frac{5(n-1)}{6}}{\binom{n}{2}} \geq \frac{2}{9}.$$

Analogously, the probability of finding a transposition that touches the fixed point $m$ and one of the circles is at least $1/n$ .

Given a transposition $a$ and a cycle $y$ of length greater than 3, we want to be able to verify that $a$ touches $y$ (i.e. that their supports are not disjoint). We need a few simple lemmas here.

**Lemma 1.** A transposition and a cycle touch if and only if they don't commute.

**Lemma 2.** A transposition $a$ and a cycle $y$ of length greater than 3 touch at only one point (i.e. their supports have only that one point in common, $a$ permutes a point inside the support of $y$ with a point outside the latter) if and only if $a$ commutes with neither $y$, nor $yay^{-1}$, nor $y^2ay^{-2}$.

Proof: Suppose the supports of $a$ and $y$ are disjoint. Then $ay = ya$.

Suppose the support of $a$ lies inside that of $y$ ($a$ permutes two points inside the cycle). Then both $yay^{-1}$ and $y^2ay^{-2}$ will permute points inside the cycle, and either $yay^{-1}$ or $y^2ay^{-2}$ will be separated from $a$ and hence will commute with $a$. Finally, if $a$ and $y$ have only one point in common, while the other point from the support of $a$ is outside the cycle, then both $yay^{-1}$ and $y^2ay^{-2}$ permute that other point from $a$ with some point from $y$ , hence neither one of the two commutes with $a$ .

**Lemma 3**. If $y_1, y_2$ are disjoint cycles of lengths $l_1, l_2$, and $a$ a permutation that touches both, then $z = y_1 a y_2$ is a cycle of length $l_1 + l_2$ . In particular, if $y$ is a cycle of length $n-1$ and $a$ is a transposition that touches $y$ at only one point, then $z = ya$ is a cycle of length $n$.

Proof is straightforward.

These lemmas show that our algorithm is correct.

## 3-4 Finding the image of an arbitrary element of $G$.

We now present an algorithm that takes as input an arbitrary element $x \in G$, elements $a, b \in G$ above, as well as the number $n$ and the set of generators of $G$, and outputs the permutation $\varpi = \Theta(x)$ which is the image of $x$ under the isomorphism $\Theta$ fixed by the condition that $\Theta(a) = (1,2)$ and $\Theta(b) = (1,\ldots,n)$. The algorithm takes $O([\omega+\mu]n\log n)$ time and requires storing $2n$ elements of $G$. The latter are pre-computed and afterwards used by each instance of the procedure.

Recall that we say that two elements of $G \cong S_n$ *touch* one another, if one of them moves (doesn't fix) a point from the support of the other, i.e. the intersection of their supports is non-empty. In simple cases, such as the case of both elements being transpositions, checking if the elements commute (or are identically equal) is enough to decide if they touch. The same is true if one element is a transposition and the other is a (single) cycle, in which case it is enough to check the commutation of the original transposition and its conjugates by the cycle and the square of the cycle.

We will write $a \sim (1,2)$, $b \sim (1,2,\ldots,n)$ instead of $\Theta(a) = (1,2)$, $\Theta(b) = (1,2,\ldots,n)$, and similarly for other elements of $G$.

As stated above, our algorithm will be nearly linear in $n$. It is based on two simple ideas. The first idea is to conjugate a known transposition (pre-computed by multiplying $a, b$) with the element $x$ and establish what transposition results from it by checking how it touches other known ones. In this fashion it is possible to establish the images of each point under $x$, i.e. the permutation $\Theta(x)$ that corresponds to $x$. Straightforward application of this idea will yield the 'naive' Method A, requiring quadratic time. The second method is to improve performance by implementing an analogue of binary search on touching permutations.

Let us now outline the 'naive' Method A, and its improvement, Method B.

**Method A.** (quadratic in $n$)

Given $a \sim (1,2)$ and $b \sim (1,2,\ldots,n)$ as above, pre-compute elements of $G$ corresponding to $a_2 \sim (2,3),\ldots,a_{n-1} \sim (n-1,n), a_n \sim (n,1)$, the transpositions of neighboring points.

Given an $x \in G$. We will subsequently find the images of the points $1,2,\ldots,n-1$ under $x$.

To find the image of 1 under $x$, compute $x^{-1}ax$. It is a transposition, and in fact, $x^{-1}ax = (x(1),x(2))$. By checking which of the $n$ pre-computed transpositions above touch it (i.e. have supports that intersect with $\{x(1),x(2)\}$), we will know the set $\{x(1),x(2)\}$. Next, compute $x^{-1}a_2x = (x(2),x(3))$, and check if it touches our pre-computed transpositions of neighboring points. Thus we will know both $x(1)$ and $x(2)$ precisely. Continue for all other points $3,\ldots,n-1$.

This process will furnish the permutation $\varpi = (i_1,i_2,\ldots,i_n)$ such that $x \sim \varpi$ in $O(n^2)$ multiplications and comparisons, $O(n)$ steps being required to find the image of each point under $x$. Pre-computation of necessary transpositions will take no more than $O(\mu n \log n)$ time even if done in the most naive way. The storage requirement is that of storing these permutations.

**Method B.** (nearly linear in $n$)

Given $a \sim (1,2)$ and $b \sim (1,2,\ldots,n)$ as above, pre-compute elements of $G$ corresponding to $a_2 \sim (2,3),\ldots,a_{n-1} \sim (n-1,n), a_n \sim (n,1)$. To find the image of a point $i$ under $x \in G$, compute $x^{-1}a_ix = (x(i),x(i+1))$. As before, the actual identity of the transposition $(x(i),x(i+1))$ is found by checking which points it touches. We now perform binary search on the set of all $n$ points on which $G$ acts, as follows.

Divide all points into two sets of lengths $k = n/2$ if $n = 2k$, or $k$ and $k+1$ if $n = 2k+1$. Likewise, divide each one of these sets in two halves, and continue recursively until you have $n$ single point sets. We have now formed the *search tree* for our search. Order the points in each set and pre-compute (starting from the elements $a$ and $b$) the cycles that correspond to each set.

For $n = 2^m$ these are the following permutations (in the cycle notation):

$$(1, 2, \ldots, \frac{n}{2}), \quad (\frac{n}{2}+1, \ldots, n),$$

$$(1, \ldots, \frac{n}{4}), \quad (\frac{n}{4}+1, \ldots, \frac{n}{2}), \quad (\frac{n}{2}+1, \ldots, \frac{3n}{4}), \quad (\frac{3n}{4}+1, \ldots, n),$$

$$\ldots$$

$$(1, 2), \quad (3, 4), \quad \ldots, \quad (n-1, n), \quad (n, 1).$$

Pre-computation of the lowest row will take soft $O(n)$ multiplications, computing the rest from the lowest row up (each time multiplying elements from the preceding row and transpositions from the lower row) will take again soft $O(n)$ multiplications. Notice that we do not need these exact cycles, only the cycles with the same supports as those in the table. We now have a binary tree of permutations.

As before, take the conjugate $x^{-1}ax = (x(1), x(2))$ and proceed with checking if this permutation touches one of the two cycles in the first row of the above table (binary tree of cycles). For the cycle(s) it does touch check their descendants in the next row below. For each row the given permutation can touch no more than two of its cycles, therefore we will know the set $\{x(1), x(2)\}$ in at most $2 \log n$ checks, i.e. $O(\log n)$ multiplications and comparisons.

Repeat this procedure with $x^{-1}a_2x$ to find both $x(1)$ and $x(2)$, and continue for $3, \ldots, n-1$. The process of finding the permutation corresponding to $x$ will take $O(n \log n)$ steps, with the requirement of simultaneously storing $2n$ elements of $G$. This requirement may be lowered at the cost of some extra multiplications.

## 3-5 Note on provable complexity.

We would like to elaborate on the nature of our algorithm's dependence on the Goldbach Conjecture. This conjecture has been verified on numerous occasions, most recently on the

Cray series, up to astronomically large [1] even numbers.

Further, it has been proved for 'almost all' even $n$ in the following sense: for large $N$ the number of even integers less than $N$ for which the conjecture does not hold is of the order $N^{1-\epsilon}$ ([MV]). There are similar results for the asymptotic behavior of $\pi(n)$, due to Hardy, Littlewood and others (see [HL], [W]). Although Hardy and Littlewood's proof of their formula in Section 3-3 assumed the General Riemann Hypothesis, it was later proved without using the latter). For the history of the Goldbach Conjecture see [W] and [Ri].

Provided that the conjecture holds, our algorithm will work for all values of $n$ , while its actual complexity will depend on the number of pairs of primes $p_1, p_2$ such that $p_1 + p_2 = n$ (even $n$) or $p_1 + p_2 = n - 1$ (odd $n$) for finding Goldbach elements and analogously for finding transpositions.

In the worst case when such pairs are few the algorithm will provably take time quadratic in $n$. However, given the conjecture, the number of such pairs is no less than $\frac{1}{3}n/\log^2 n$, hence the complexity of the algorithm will be $O([\rho + \mu + \omega]n \log^2 n)$ rather than quadratic.

It has been demonstrated that any large enough number can be represented as a sum of three primes (see [V], [W]). If we modify our algorithm so that instead of Goldbach elements of cycle forms $(p_1, p_2)$ it uses elements of cycle forms $(p_1, p_2, p_3)$ , we get an algorithm which, at worst, requires *provably* cubic time, while for general $n$ with numerous representations $n = p_1 + p_2 + p_3$ it may again require only nearly linear time.

Going one step further, for six primes it *is* known (see [Ra]) that for all even $n > 10$ there is a presentation $n = p_1 + \ldots + p_6$ . Consequently, another modification of our algorithm will give an algorithm *provably* succeeding in $O(n^6)$ time.

Finally, we can combine these arguments into a single algorithm, in which for the given $n$ we find its representation as a sum of a number of distinct primes, thus locating an element consisting of primes cycles of different lengths (Goldbach elements are a a specific case of these).

[1] Deshouillers, Yannick Saouter and Herman de Riele have verified this up to $10^{14}$ with the help, of a Cray C90 and various workstations. See [Ri] and [W] for more information.

Using these elements instead of only the Goldbach elements, we can expect a somewhat faster algorithm. However, for the sake of brevity, we will not go into details of this, since the latter should not pose any significant conceptual problems over the presented algorithm.

To summarize, we have chosen the venue which is at the same time efficient and easy to present, although not formally proven as mathematical purists will require.

Concluding our discussion of complexity, we would like to point out that it seems to be impossible to construct a Las Vegas recognition algorithm for a gray box group $G$ isomorphic to $S_n$ that would require $o(n)$ time. The reason for this lies in the combinatorial structure of the symmetric group. To solve the recognition problem we need to find a number of generators which fix an isomorphism $\Theta : G \to S_n$. Suppose we are lucky enough to find these elements really fast, but to make sure that they are indeed the generators we take them for, we need to verify certain relations between them. However, for all known presentations of $S_n$ the minimal number of defining relations is at least linear. For example, in our case, for a transposition and a long cycle we need to verify at least $n/2$ relations (see [CM], §6.2). Moreover, Coxeter and Moser conjectured that the latter bound cannot be improved.

## 3-6 Modification for the alternating group $A_n$.

It turns out that the Goldbach conjecture approach can be applied to the recognition of a gray box group $G$ known to be isomorphic to $A_n$. Indeed, $A_n$ for $n$ odd is generated by a cycle of length 3 and a long cycle. For all odd primes $p_1, p_2$ the corresponding Golbach elements of $S_n$ actually lie inside $A_n$. Therefore we can first locate a Goldbach element in $A_n$ and break it down into two prime cycles by raising it to corresponding powers.

Then we find a 3-cycle using a technique similar to that of Section 3-2.1. Once a 3-cycle is found, we can obtain any number of (nearly) uniformly distributed random 3-cycles by conjugating it with random elements of the group $G$. We then find a 3-cycle that touches both cycles of the Golbach element. The product of these and the found 3-cycle gives us a long cycle in $G$. Re-numbering the points accordingly, we arrive at two elements $a, b \in G$ such that there is an isomorphism $\psi : G \to A_n$ carrying $a$ to $(1, 2, 3)$ and $b$ to $(1, 2, \ldots, n)$.

The case of $G \cong A_{2k}$ involves some additional technical details. We omit the details.

## 3-7 What to do if $n$ is not known?

In this section we present a Las Vegas algorithm for verifying that a black box group $G$ is isomorphic to $S_n$ for a known $n$. We then give a Monte Carlo algorithms that finds $n$ when it is not known a priori. The combination of these two algorithms applied to a black box group $G$ is Las Vegas, i.e. a positive answer and the resulting recognition are guaranteed to be correct, while a negative answer may be incorrect, with probability of error $\varepsilon > 0$ that can be made as small as desired.

Assume that $G \cong S_n$, and start, as before, the search for a Goldbach element. If no Goldbach element is found after a sufficient number $N(\varepsilon, n)$ (discussed below) of random elements have been tried, the algorithm stops and reports a negative answer. The same happens during the search for a transposition. If both searches are successful, we have elements $a, b \in G$.

Check if $a$ and $b$ satisfy the Moore's defining relations for $S_n$ (see [CM]), namely

$$b^n = a^2 = (ba)^{n-1} = (ab^{-1}ab)^3 = (ab^{-j}ab^j)^2 = id, \qquad 2 \le j \le n-2.$$

This can be done in $O(\mu n)$ time. If this is the case, the mapping $\psi : S_n \to G$ defined by $\psi(1\ 2) = a$, $\psi(1\ 2 \ldots\ n) = b$ defines a homomorphism of $S_n$ onto its image $G_0 = \langle a, b\rangle \subset G$. Being a homomorphic image of $S_n$, $G_0$ can only be one of $S_n$, $Z_2$ or $\{id\}$, and excluding the latter two cases for $G = \langle a, b\rangle$ is trivial. Therefore $G_0 \cong S_n$.

To demonstrate $G \cong S_n$ we need to check that $G_0 = G$. Given $G = \langle g_1, \ldots, g_s\rangle$, we need to express the original generators $g_1, \ldots, g_s$ as words in $a, b$. Notice that the cycle tree used in the second phase of our algorithm is constructed entirely inside $G_0$. Running the second phase for $g_i$, $i = 1, \dots, s$ we obtain some permutations $\gamma_i \in S_n$, $i = 1, \dots, s$. Represent these permutations as words in the generators $(1\ 2)$, $(1\ 2\ \dots\ n)$ of $S_n$ and construct the images $\psi(\gamma_i)$, $i = 1, \dots, s$. If $\psi(\gamma_i) = g_i$, we have the desired isomorphism, if not, we can claim that $S_n \cong G_0 \neq G$. This step of verification procedure takes $O(\mu\, s\, n \log n)$ time.

Let us estimate the number $N(\varepsilon, n)$ of trials that fail to produce a Goldbach element, after which we can claim, with probability of error less than $\varepsilon$, that the group $G$ is not isomorphic to $S_n$. We need at most

$$N(\varepsilon, n) = 2 \left\lceil \log \frac{1}{\varepsilon} \right\rceil \frac{1}{P_n}$$

trials. Indeed, if $G \cong S_n$, then after $2/P_n$ trials the probability of finding a Goldbach element is at least $1/2$. Therefore after $2C/P_n$ trials the latter probability is at least $1 - 1/2^C$. Taking $C = \lceil \log \frac{1}{\varepsilon} \rceil$ proves the claim. A similar estimate holds for the number of unsuccessful trials while searching for a transposition in $G$. Using the estimate of $P_n$ the extended Goldbach conjecture 3-1.3, we get the following result.

**Theorem 3-7.1** *Let $G = \langle g_1, \dots g_s \rangle$ be a black box group. Let $\mu$ be the time required to perform a group operation in $G$, $\rho$ the time required to compute a (nearly) uniform random element of $G$, let $\omega$ be the time required to find the order of an arbitrary element of $G$. Then, provided that Conjecture 3-1.3 holds, a Las Vegas algorithm exists, which in time*

$$O\left( [\rho + \mu + \omega] \log \left( \frac{1}{\varepsilon} \right) n \log^2 n + \mu\, s\, n \log n \right)$$

*either proves that $G$ is isomorphic to $S_n$ or rejects this hypothesis with probability of error $\varepsilon$.*

We now present a Monte Carlo algorithm that, given a black box group $G$ isomorphic to $S_n$ for some $n$ *not known a priori*, finds $n$. This value of $n$ can be verified using the Las Vegas algorithm above.

We assume that there exists a natural upper bound for $n$. Such an upper bound arises when the elements of the black box group are encoded with bit strings of equal (finite) length $M$.

In that case we know that $n < N := f(2^M)$, where $f$ is the inverse factorial. For simplicity assume that $N$ is even.

Denote

$$P_n = \sum_{p>q>2 \text{ primes}\,,\, p+q=n} \frac{1}{p\,q}.$$

Clearly, $P_n$ estimates the proportion of Goldbach elements in $S_n$. Denote by $\kappa(N)$ the maximum of $1/P_{2i}$ over all $i = 1, \ldots, N/2$ . Take a sequence of $2\kappa(N)$ random elements of $G$, and for every random element $x$ compute its order

$$|x| = p_1^{k_1} \cdot p_2^{k_2} \cdot \ \ldots \ \cdot p_m^{k_m}.$$

Take the maximum of sums

$$S = p_1 + p_2 + \ \ldots \ + p_m.$$

over all these $x$. It is easy to see that $S \le n$, and equals $n$ exactly when the element $x$ consists of cycles of distinct prime lengths $p_1, p_2, \ldots, p_m$ (and no fixed points). In particular, if $x$ is a Goldbach element and $n$ is even, we get the equality, and if $n$ is odd then $S = n - 1$. We claim that after $2\kappa(N)$ steps the probability of getting $S = n$ or $S = n - 1$ is at least $1/2$. Indeed, the expected number of steps before a Goldbach element is encountered is at most $1/P_n$ when $n$ is even, or $1/P_{n-1}$, when $n$ is odd. This, and the Markov bound immediately imply the above estimate.

Deciding whether $S$, the estimated degree of the group, is equal to $n$ or $n - 1$ requires more time and involves looking for elements of order $3p_1p_2$, where $p_1 + p_2 + 3 = S + 1$. If no such elements were found after $6\kappa(S)$ steps, we can conclude, with the probability of error $< 1/2$ that $S = n$. Repeating this $k$ times will improve the probability of error to $1/2^k$.

It only remains to recall that, by the extended Goldbach conjecture, we have $\kappa(N) = O(N \log^2 N)$. So the following statement holds.

**Theorem 3-7.2** *In the notations of the previous theorem, given an upper bound $N$ for $n$, and* *provided that Conjecture 3-1.3 holds, a Monte Carlo algorithm exists which in time* $O([\rho + \omega + \mu] N \log^2 N)$ *can find* $n$.

# Chapter 4

# Recognition of black box groups isomorphic to $PSL(n,q)$ or perfect central extensions of $PSL(n,q)$

## 4-1 Main Theorem

In this chapter we present an elementary Las Vegas algorithm for the constructive recognition of $SL(n,q)$, or more generally, any perfect central extension $G$ of $PSL(n,q)$, which is represented as a *black box group*. In this model, the group elements are encoded by binary strings of uniform length $N$, and group operations are performed by an oracle (the black box). The oracle can compute the product of elements, the inverse and recognize the identity element in time polynomial in $N$.

The approach presented here is motivated in part by earlier work [CFL] which provided the first constructive recognition of an infinite family of classical groups presented in a black box setting which could be accomplished in time polynomial in the input length $N$, the time for a black box group operation $\mu$ and the time $\rho$ required to compute a (nearly) uniform random element of $G$. The present thesis extends the result of [CFL] from the family $SL(n, 2)$, $n \geq 3$, to the family $SL(n, q)$ over any finite field $GF(q)$, $q \geq 3$, and to any other perfect central extensions of $PSL(n, q)$. Recall that a *central extension* of a group $H$ by an abelian group $A$ is a group $E$ such that there is a short exact sequence of groups

$$\{id\} \to A \xrightarrow{i} E \xrightarrow{\pi} H \to \{id\}$$

such that $i(A)$ is central in $E$ (equivalently, the action of $H \cong E/A$ on $A$, induced by the conjugation action of $E$ on its normal embedded subgroup $i(A)$, is trivial). A group $H$ is called *perfect* if $H = [H, H]$. In what follows, we use the known fact that for $n > 3$, $q > 4$ any perfect central extension of $PSL(n, q)$ is isomorphic to a factor group of $SL(n, q)$ by some central subgroup $Z_0 \subset \mathcal{Z}(SL(n, q))$.

Our main result is as follows:

**Theorem 4-1.1** *Let $G$ be a black box group specified by a generating set $\mathcal{G}$ with each element of $G$ represented by a bit string of length $N$. There is a Las Vegas algorithm for verifying that $G$ is isomorphic to a central extension of $PSL(n, q)$ for a specified $n$ and $q$. The algorithm has input ($\mathcal{G}$, $n$, $q$) and takes time*

$$O(|\mathcal{G}|N + n^3 \log n (\log q)^4 + nq\rho + n^3\epsilon + n^2\mu q \log q)$$

*where $\mu$ is the time required to perform a group operation in $G$, $\rho$ is the time required to compute a (nearly) uniform random element of $G$, and $\epsilon$ is the time for a field operation in $GF(q)$. If $G$ has the required isomorphism type, then the algorithm will confirm this with probability greater than $1/2$ and produce data structures for deterministically computing:*

- *the order of the center of $G$, and hence the isomorphism type of $G$,*
- *a homomorphism $\Theta : SL(n, q) \longrightarrow G$, in which the time to compute $\Theta(\xi) \in G$ takes time $O(n^3\varepsilon + n^2\mu \log q)$ and the time to find $\xi \in \Theta^{-1}(x)$, $x \in G$ takes time $O(N + n^2 q\mu)$.*

*Otherwise, the algorithm will return failure.*

## 4-2 Transvections in $SL(n,q)$ and $PSL(n,q)$

Transvections of $SL(n,q)$ and $PSL(n,q)$ play a key role in our algorithm. A *transvection* $\tau$ in $SL(n,q)$ is an element of $SL(n,q)$ which can be identified with a vector $v \in V$ and a linear functional $f \in V^*$ such that $f(v)=0$ (here and henceforth we always imply that $v \neq 0$, $f \neq 0$) and $\tau$ acts on $V$ according to the following rule:

$$w\tau = w + f(w)v, \quad \forall\, w \in V,$$

where $V$ is the space of the natural matrix representation of $SL(n,q)$ and $GL(n,q)$. It can be easily checked that, in fact, for any $\tau' \in GL(n,q)$ that acts on $V$ as above, $\tau' \in SL(n,q)$. We refer to $v$ as the *center* of the transvection $\tau$, and to the hyperplane $Ker(f)$ as the *axis* of $\tau$. Note that $\tau$ acts as the identity on its axis. The notation we adopt for representing the dependency of $\tau$ on $v$ and $f$, needs to reflect the fact that for arbitrary $\lambda \in GF(q)$, simultaneously replacing $v$ by $\lambda v$ and $f$ by $\lambda^{-1} f$ does not not change $\tau$. Accordingly, we denote the transvection $\tau$ by $\tau_{v\otimes f}$. This notation is further justified by observing that under the canonical bijection between $L(V) \supset End(V)$ and $V \otimes V^*$, which identifies any element $z = \sum v_k \otimes f_k \in V \otimes V^*$ with the operator

$$M_z \ : \ w \ \mapsto \ \sum f_k(w)\, v_k \ , \qquad \forall w \in V \ ,$$

the set of transvections of $SL(n,q)$ can be identified with the subset of $V \otimes V^*$ of the form $I + v \otimes f$, where $v \in V$, $f \in V^*$ and $f(v)=0$. More precisely, we have a well-defined one-to-one map between

$$\{I + v \otimes f \mid f(v) = 0\} \subset V \otimes V^*$$

and

$$\{\tau \in End(V) \mid \exists v, f \ \text{ s.t. } \ w\tau = w + f(w)v \quad \forall w \in V\}$$

given by

$$( \tau \ \text{ s.t. } \ \forall w \in V \ w\tau = w + f(w)v \ ) \quad \mapsto \quad (I + v \otimes f) \ .$$

It is easy to see that each coset in $SL(n,q)$ of its center $\mathcal{Z}(SL(n,q)) = \{\alpha \, : \, \alpha^n = 1\}$, either contains a unique transvection or none at all. We will therefore refer to transvections as either elements of $SL(n,q)$ in the standard sense, or any coset of a subgroup $Z_0 \subseteq \mathcal{Z}(SL(n,q))$ which contains a transvection. In the latter case we take that unique transvection as a representative of its coset in $SL(n,q)/Z_0$. This allows us to speak of transvections in the quotient group $SL(n,q)/Z_0$. In particular, we will speak of transvections in $PSL(n,q) = SL(n,q)/\mathcal{Z}(SL(n,q))$. Clearly, there is a natural bijection between the sets of transvections of $SL(n,q)$ and $PSL(n,q)$, or between the set of transvection of $SL(n,q)$ and that of any perfect central extension $G$ of $PSL(n,q)$. Under any epimorphism $\Theta : SL(n,q) \to G$ transvections are mapped to transvections.

Let $SL(n,q)$, or $PSL(n,q)$, act on itself by conjugation. Under conjugation transvections go to transvections, and the actions that both groups induce on the sets of their transvections are the same. Moreover, these actions naturally correspond to the linear representation of $PSL(n,q)$ on $V \otimes V^*$, as follows.

Consider the action of $SL(n,q)$ on $V \otimes V^*$ defined by

$$\xi(v \otimes f) = v\xi \otimes \xi^* f \ ,$$

where the action on $V^*$ is the adjoint action, $(\xi^* f)(\,\cdot\,) = f(\,\cdot\,\xi)$. This action *factors through* $PSL(n,q)$, since any $x$ and $\lambda x$ ($\lambda \in GF(q)$) are indistinguishable by this action. Note that for $x \in PSL(n,q)$ its action on either $V$ or $V^*$ is no longer defined, but we can still define its action on $V \otimes V^*$. Namely, let $\psi : SL(n,q) \to PSL(n,q)$ be the canonical epimorphism. Take $x \in PSL(n,q)$ and a coset representative $\xi \in \psi^{-1}(x)$, and define

$$x(v \otimes f) = v\xi \otimes \xi^* f \ , \qquad \xi \in \psi^{-1}(x) \ .$$

Clearly, this action is well-defined. It naturally extends to any perfect central extension $G$ of $PSL(n,q)$ with the canonical epimorphism $\psi' : G \to PSL(n,q)$, and an epimorphism $\Theta$ : $SL(n,q) \to G$, giving rise to the following commutative diagram of actions derived from the above action of $SL(n,q)$ on $V \otimes V^*$ :

$$\begin{array}{ccc} SL(n,q) & & \\ \Theta\downarrow & \searrow & \\ G & \longrightarrow & \mathrm{End}(V \otimes V^*) \\ \psi'\downarrow & \nearrow & \\ PSL(n,q) & & \end{array}$$

The key fact that ties these actions of $SL(n,q)$, $PSL(n,q)$, or of any perfect central extension $G$, on $V \otimes V^*$ with their respective actions by conjugation on transvections is that

$$\tau^{\xi}_{v\otimes f} = \tau_{\xi(v\otimes f)} = \tau_{v\xi\otimes\xi^* f}$$

for any $\xi \in SL(n,q)$, or, for any perfect central extension $G$ of $PSL(n,q)$, $x \in G$,

$$\tau^{x}_{v\otimes f} = \tau_{x(v\otimes f)} = \tau_{v\xi\otimes\xi^* f}, \qquad x = \Theta(\xi)$$

where the choice of $\xi \in \Theta^{-1}(x) \subset SL(n,q)$ can be arbitrary.

Thus the action of $SL(n,q)$ (respectively, $PSL(n,q)$ and $G$) on their sets of transvections coincides with its action on the set

$$\{v \otimes f \mid f(v) = 0\} \subset V \otimes V^* .$$

Notice that in $SL(n,q)$ or in any central extension of $PSL(n,q)$, the center of the group is exactly the kernel of this action, so the effective action on transvections in such a group is that of $PSL(n,q)$, and this latter action is faithful.

## 4-3 Overview of the Algorithm

Let $G$ be a black box group which is known to be a perfect central extension of $PSL(n,q)$, for known $n$ and $q$ that serve as an input to the algorithm, together with the generating set $\mathcal{G}$ of $G$. We want to construct an epimorphism $\Theta : SL(n,q) \to G$. For each element $x \in G$ represented as a bit string of length $N$, we would like our algorithm to output an $n \times n$ matrix with entries in $GF(q)$, which corresponds to an element $\xi \in SL(n,q)$ such that $x = \Theta(\xi)$. More technically, let $p$ be the characteristic of $GF(q)$, $q = p^l$. The algorithm will return, for $x \in G$, an $n \times n$ matrix with entries in $GF(q)$ represented as $l$-tuples $(b_0, \ldots, b_{l-1})$, $b_k \in \mathbb{F}_p$.

The homomorphism $\Theta$ will be described by the images in $G$ of certain elementary matrices that generate $SL(n,q)$. We will find these images one by one, starting from a single element of $G$ for which we can verify that it is a transvection. When the images of the chosen system of generators of $SL(n,q)$ are constructed in $G$ (Sections 2–6), we verify that the subgroup $G_0 \subset G$ generated by them is in fact a homomorphic image of $SL(n,q)$. This is done by checking a version of Steinberg-Curtis-Tits relations (see [BGKLP]). At this point we can affirm that $\Theta : SL(n,q) \to G_0$ is in fact an epimorphism.

We then make sure that the subgroup $G_0$ coincides with $G$, by verifying that each generator of $G$ lies in $G_0$. In the course of this procedure we construct the pre-image in $SL(n,q)$ of the center of $G$ and develop an efficient algorithm for computing $\Theta(\xi)$, $\xi \in SL(n,q)$. After such verification we can affirm that we have indeed constructed an epimorphism $\Theta : SL(n,q) \to G$, that is, our algorithm is Las Vegas.

The above mentioned representation of elements of $GF(q) = GF(p^l)$ is, of course, the standard representation of a finite field by polynomials modulo an irreducible polynomial. More specifically, fix an element $\theta \in GF(q)$ which lies in no proper subfield of $GF(q)$. Then $GF(q) \cong \mathbb{F}_p[x]/(f_\theta(x))$, where $f_\theta(x)$ is the minimal polynomial of $\theta$. Elements of the field

$GF(q)$ are represented by $l$-tuples $(b_0, b_1, \ldots, b_{l-1})$, $b_i \in \mathbb{F}_p$, so that

$$(b_0, b_1, \ldots, b_{l-1}) \longleftrightarrow b_0 + b_1\theta + \ldots + b_{l-1}\theta^{l-1}.$$

We use this model of $GF(q)$ for all computations in the field (when $q$ is a prime-power; for prime $q$ the model is much simpler).

From now on, $G$ will refer to the black box group, which is known to be a perfect central extension of $PSL(n,q)$. The desired homomorphism is denoted $\Theta : SL(n,q) \to G$. Denote by $Z = \mathcal{Z}(G)$ the center of $G$ and by $Z_0 = \Theta^{-1}(Z)$ its pre-image in $SL(n,q)$. The elements of $G$ are denoted by Latin letters $(x, s, t, \ldots)$, and the elements of $SL(n,q)$ by Greek letters $(\xi, \sigma, \tau, \ldots)$. $V$ will denote an $n$-dimensional vector space over $GF(q)$, and $V^*$ the dual space $Hom_{GF(q)}(V, GF(q))$ of $GF(q)$-linear homomorphisms from $V$ to $GF(q)$ (linear functionals from $V$ to $GF(q)$). Vectors from $V$ are usually denoted by $u, v, \ldots$ and functionals from $V^*$ by $f, g, h, \ldots$. For example, a transvection in the matrix group $SL(n,q)$ defined by a vector $v \in V$ and a functional $f \in V^*$ is denoted by $\tau_{v\otimes f}$, while its image in the black box group $G$ under $\Theta$ is denoted by $t_{v\otimes f}$ (this last bit of notation is further explained below).

The construction of $\Theta$ depends on three critical properties of transvections in $SL(n,q)$.

**(1)** Given transvections, $\tau_{v_1\otimes f_1}, \tau_{v_2\otimes f_2}$, one can determine certain relations between $v_1, v_2, f_1, f_2$ by a simple "straight line program" without specific knowledge of $v_1, v_2, f_1, f_2$. In particular, not only can we check whether $f_i(v_j) = 0$, but also obtain equations from which $f_1(v_2) \cdot f_2(v_1)$ can be found explicitly.

**(2)** The conjugation action of $SL(n,q)$ on its set of transvections is given by

$$\tau^{\xi}_{v\otimes f} = \tau_{\xi(v\otimes f)} = \tau_{v\xi\otimes\xi^* f}.$$

Any perfect central extension of $PSL(n,q)$ possesses a derived action, which is defined via an epimorphism from $SL(n,q)$. All of these actions effectively correspond to the linear representation of $PSL(n,q)$ on $V \otimes V^*$.

**(3)** The permutation representation of $SL(n,q)$ acting on the set of all transvections by conjugation is imprimitive with precisely two maximal systems of imprimitivity. The representatives of these block systems which contain $\tau_{v\otimes f}$ can be described as the sets $\Gamma(f) = \{\tau_{u\otimes f} \mid f(u) = 0,\ u \neq 0\}$ and $\beta(v) = \{\tau_{v\otimes h} \mid h(v) = 0,\ h \neq 0\}$. These sets have simple geometric interpretations: $\Gamma(f)$ consists of all transvections with the common axis $Ker(f)$, and $\beta(v)$ consists of all transvections with the common center $v$. The intersection of these two blocks of imprimitivity is the *full transvection group*

$$\beta(v) \cap \Gamma(f) = \{\tau_{\alpha v\otimes f} \mid \alpha \in GF(q)\}$$

of $\tau_{v\otimes f}$, which we use as the computational model of the field $GF(q)$.

The architecture of the algorithm can be outlined as follows. Each stage will be dealt with in detail later.

**(i)** Find a transvection $t \in G$ by sampling random elements until an element with the right properties is found. Simultaneously, find a $ppd(q, n-2)$ element $s \in G$ that commutes with $t$, which is used later in (iii) for constructing the "dual block pair".

**(ii)** Construct the full transvection group $T_{v\otimes f}$ of the transvection $t = t_{v\otimes f}$ found in (i). In the course of this procedure, a basis of this full transvection group is found. $T_{v\otimes f}$ is used as a computational model of the field.

**(iii)** Construct a basis for the dual block pair of $t_{v\otimes f}$, which consists of $2n-2$ transvections. For each transvection in this basis, construct a basis of its full transvection group. This gives a set of $2(n-1)l$ transvections that generate a subgroup $G_0 \in G$. Also, the choice of the basis for the dual block pair fixes a basis in $V$, which is used for writing matrices over the finite field.

**(iv)** Define $\Theta : SL(n,q) \to G$ as a mapping that takes elementary matrices to elements of the constructed dual block pair basis. Check the reduced Steinberg–Curtis–Tits commutation relations to prove that the subgroup $G_0$ is indeed a homomorphic image of $SL(n, q)$.

**(v)** Develop an algorithm for constructing the image $\Theta(\chi)$ of a $\chi \in SL(n, q)$ defined by its matrix $M_\chi$.

**(vi)** Using (v), construct the image in $G$ under $\Theta$ of the center of $SL(n, q)$. This determines the isomorphism type of $G$, i.e. finds if $G \cong PSL(n, q)$ or, if $G$ is a perfect central extension of $PSL(n, q)$, determines its type up to an isomorphic equivalence.

**(vii)** Develop an algorithm for constructing a representative of $\Theta^{-1}(x)$. For this purpose we use the conjugation action of $x \in G$ on the basis of the dual block pair.

**(viii)** Using (vii), prove that all generators of $G$ lie in $G_0$, i.e. that $G = G_0$ and $\Theta$ is an epimorphism. This completes the recognition. The pre-images of individual elements of $G$ are found by (vii).

Recall that for any transvection $t \in G$, $\Theta^{-1}(t)$ contains a unique transvection, some $\tau_{u\otimes g}$ defined by $u \in V$, $g \in V^*$. It will be convenient to denote the image of $\tau_{u\otimes g}$ by $t_{u\otimes g}$, $t_{u\otimes g} = \Theta(\tau_{u\otimes g})$, thus relating the labeling of transvections in $G$ and $SL(n, q)$. On each step of our construction of $\Theta$ we will make sure that the labeling of transvections in $G$ and $SL(n, q)$ is consistent with $\Theta$.

Our computations require that we construct a model of the finite field $GF(q)$ This model is trivial when $q = p$ for a prime $p$. For $q = p^l$, $l > 1$ we use the elementary abelian subgroup of $G$ which is constructed as the image under $\Theta$ of the full transvection group $\{\tau_{\alpha v\otimes f} : \alpha \in GF(q)\} \subset SL(n, q)$. We construct this subgroup starting with the original transvection $t_{v\otimes f}$. Using commutation relations of transvections and conjugation by random elements of $G$, we construct the set

$$T = \{t_{v\otimes f}, t_{\theta v\otimes f}, t_{\theta^2 v\otimes f}, \ldots, t_{\theta^{l-1} v\otimes f}\},$$

which is a set of generators for the elementary abelian subgroup $T_{v\otimes f} \subset G$ such that

$$T_{v\otimes f} = \Theta(\{\tau_{\alpha v\otimes f} \mid \alpha \in GF(q)\}) \ ,$$

where $\theta$ lies in no proper subfield of $GF(q)$.

The algorithm used to construct $t_{\theta v\otimes f}$ starting from $t_{v\otimes f}$ guarantees that the element obtained is of the said form (label), and the remaining elements of $G$ are constructed so that the labeling remains consistent with $\Theta$. In other words, we can get any $\theta \in GF(q)$, i.e. any transvection of the form $t_{\theta v\otimes f}$, with equal probability. But then the group generated by $T$ is elementary abelian and isomorphic to a subfield of $GF(q)$, so we can find the minimal polynomial of $\theta$ and establish its identity right away (up to a field automorphism).

For any transvection $t_{u\otimes g}$ obtained by conjugation or commutation from $t_{v\otimes f}$ (all transvections we use are obtained that way from the original one) we can construct the generating set of $T_{u\otimes g}$ of the same form with the same $\theta \in GF(q)$.

Given the transvection $t_{v\otimes f} \in G$, we successively use properties (1)-(3) to construct subsets $B_\Gamma$ and $B_\beta$ of $G$ (see section 4-7.1) which we will refer to as a *dual block pair with respect to* $v \otimes f$ and which satisfy the following properties:

- $B_\Gamma = \Theta(\{\tau_{v_1\otimes f}, \tau_{v_2\otimes f}, \ldots, \tau_{v_{n-1}\otimes f}\})$ where $\{v_1, \ldots, v_{n-1}\}$ is a basis for $Ker(f) = \{w \in V \mid f(w) = 0\}$.

- $B_\beta = \Theta(\{\tau_{v_n\otimes g_1}, \ldots, \tau_{v_n\otimes g_{n-1}}\})$ where $f(v_n) = 1$ and $\{g_1, \ldots, g_{n-1}\}$ is a basis for $(v_n)^\perp = \{g \in V^* \mid g(v_n) = 0\}$.

- $g_i(v_j) = \delta_{ij}$, $1 \le i, j \le n-1$, where $\delta_{ij}$ is interpreted as an element of $GF(q)$.

In the above we define $v_1 := v$ and $g_n := f$ so that $t = \Theta(\tau_{v_1\otimes g_n})$, where $t$ is the original transvection in $G$, and $\{g_1, \ldots, g_n\}$ is a basis for $V^*$ dual to the basis $\{v_1, \ldots, v_n\}$ of $V$.

The construction of $B_\Gamma$ and $B_\beta$ is presented in section 4-7.1. We denote

$$B_\Gamma = \{t_{v_1\otimes g_n}, \ldots, t_{v_{n-1}\otimes g_n}\} \ ,$$

and

$$B_\beta = \{t_{v_n\otimes g_1}, \ldots, t_{v_n\otimes g_{n-1}}\}\ .$$

For each transvection $t_{v_i\otimes g_j} \in B_\Gamma \cup B_\beta$ we construct the generators

$$\{t_{v_i\otimes g_j}, t_{\theta v_i\otimes g_j}, t_{\theta^2 v_i\otimes g_j}, \ldots, t_{\theta^{l-1} v_i\otimes g_j}\},$$

of its full transvection group. Together they form the system

$$S' = \{t_{\theta^k v_i\otimes g_j} \mid t_{v_i\otimes g_j} \in B_\Gamma \cup B_\beta,\ k = 0, \ldots, l-1\}\ ,$$

for which we check a version of Steinberg-Curtis-Tits relations to verify that $\Theta : SL(n,q) \to G_0 = \langle S' \rangle$ defined by $t_{\theta^k v_i\otimes g_j} := \Theta(\tau_{\theta^k v_i\otimes g_j})$ is indeed an epimorphism. We then show how to construct the image $\Theta(\chi) \in G$ of an arbitrary element $\chi \in SL(n,q)$ given by its matrix $M_\chi$. In particular, we use this method to compute (element-wise) the image of the center $\mathcal{Z}(SL(n,q))$, which is the center $Z = \mathcal{Z}(G)$ of $G$.

Given $x \in G_0$, $\xi \in \Theta^{-1}(x)$ can be found by setting up a set of equations describing the action of $x$ on $B_\Gamma \cup B_\beta$. These equations are solved for the coefficients of the matrix of $\xi$ relative to the basis $\{v_1, \ldots, v_n\}$. More precisely, the action of $x$ on $B_\Gamma \cup B_\beta$ implies its action on $V \otimes V^*$, in particular, on its standard basis $\{v_i \otimes g_j\}$ represented by transvections from $B_\Gamma \cup B_\beta$. The coefficients of the matrix $M_\xi$ are recovered, up to a fixed scalar constant, from $2n-1$ equations describing the latter action.

Finally, to verify that $G = G_0$, we obtain and solve $2\,s(n-1)$ equations for the original generators $\langle g_1, \ldots, g_s\rangle = G$. We get the matrices $M_1, \ldots, M_s$ of some elements $\gamma_1, \ldots, \gamma_s \in SL(n,q)$. To verify that $G_0 = G$ (so that $\Theta$ is indeed an epimorphism) we only need to construct $\Theta(\gamma_1), \ldots, \Theta(\gamma_s)$ and verify that $\Theta(\gamma_i) = g_i$, $i = 1, \ldots, s$, which completes the Las Vegas recognition of $G$.

Throughout the algorithm we need to produce (nearly) uniformly distributed random elements of the black box group $G$. We assume that an efficient algorithm for producing such elements is available; see Lazlo Babai [Ba1], [Ba2] for details.

## 4-4 Finding a transvection in $G$

In this section we describe a Las Vegas algorithm for finding a transvection of $G$ together with a commuting element that acts irreducibly on a hyper-hyperplane (i.e. a linear subspace of dimension $n-2$) of the underlying vector space $V$. The algorithm is Las Vegas under the assumption that $n$ and $q$ are known. The basic idea is to first derive certain properties of transvections in $SL(n,q)$ and $PSL(n,q)$ which are numerical in nature. These properties are independent of the representation of $SL(n,q)$ or $PSL(n,q)$ and hence can be applied to the black box group $G$.

We will obtain a transvection as a result of a series of random trials of independent uniformly distributed elements of $G$. The information necessary to isolate a transvection is contained in the orders of these elements. Recall that

$$|SL(n,q)| = q^{\frac{n^2-n}{2}}(q^n-1)(q^{n-1}-1)(q^{n-2}-1)\dots(q^2-1)\,,$$

that is, the divisors of $|SL(n,q)|$ (and the orders of its elements) are those of $q^i-1$ for $i=2,\dots,n$. An element $x$ whose order is known to contain a divisor of $q^n-1$ but not of any $q^i-1$ for $i=2,\dots,n-1$ is easily seen to act irreducibly under the natural representation of $SL(n,q)$. Analogously, an element $y\in G$ such that $|y|$ is known to contain a divisor of $q^k-1$ but not of $q^i-1$ for $i<k<n$, must act irreducibly on any invariant subspace of dimension $k$. The following result of Zsigmondy makes this observation useful for all $SL(n,q)$, $n>3$, $q>4$:

**Lemma 4-4.1** *(Zsigmondy, 1892). If $a,d\in\mathbb{Z}$, $a\geq 2$, $\quad d\geq 3$, and the pair $(a,d)\neq(2,6)$, then there is a prime which divides $a^d-1$ but does not divide $a^i-1$ for all $1\leq i\leq d-1$.*

**Definition 4-4.2** *A prime $p'$ is said to be a primitive prime divisor of $q^k-1$ for $(q,k)$ if $p'\mid(q^k-1)$ but $p'$ does not divide any $(q^j-1)$ for $j<k$. An element $\sigma$ of $GL(n,q)$ is said to be a ppd$(n,q,e)$ element, $n/2<e\leq n$, if $\sigma$ satisfies two conditions. First, it should leave invariant an $(n-e)$ dimensional subspace of the underlying vector space. Second, $|\sigma|$ contains*

*a primitive prime divisor of* $q^e - 1$ *for* $(q, e)$ *when* $(q, e) \neq (2, 6)$*, and otherwise,* $|\sigma| = 63$*. We can extend this definition to* $PSL(n, q)$ *or to any perfect central extension of* $PSL(n, q)$ *saying that the coset* $\bar{x} = x\mathcal{Z}(SL(n, q))$ *is a ppd*$(n, q, e)$ *element if* $\bar{x}$ *contains a ppd*$(n, q, e)$ *element of* $SL(n, q)$*.*

**Proposition 4-4.3** *Let elements* $\tau$ *and* $\sigma$ *be commuting elements of* $SL(n, q)$*,* $n \geq 3$*,* $q = p^l$*, such that* $|\tau| = p$ *and* $|\sigma| = p'$*, where* $p$ *is the characteristic of* $GF(q)$*, and* $p'$ *is a primitive prime divisor of* $q^{n-2} - 1$ *for* $(q, n-2)$*. Then* $\tau$ *is a transvection and* $\sigma$ *acts irreducibly on an* $(n-2)$*-dimensional subspace of* $C_V(\tau) := \{v \in V \mid v\tau = v\}$*, the hyperplane which is the center of* $\tau$*.*

We require the following supporting result.

**Lemma 4-4.4** *Given the notation of Proposition 4-4.3, if* $|\sigma| = p'$*, then all representations of* $\langle\sigma\rangle$ *of dimension less than* $n-2$ *over a field with characteristic* $p$ *are trivial, and any non-trivial representation of dimension* $n-2$ *is irreducible.*

*Proof:* For any representation $\rho : \langle\sigma\rangle \to End(W)$, where $W$ is a linear space over $GF(q)$ , we have $\ker\rho = \{e\}$ or $\ker\rho = \langle\sigma\rangle$, since the cyclic group $\langle\sigma\rangle$ is a group of prime order and thus has no proper subgroups. Therefore $\rho$ is either faithful or trivial. If $\rho$ is faithful, then $|\sigma|$ divides

$$|SL(W)| = (q^m - 1)(q^m - q)\dots(q^m - q^{m-1}) = (q^m - 1)(q^{m-1} - 1)\dots(q-1)q^{\frac{m(m-1)}{2}} ,$$

which implies that $m = \dim(W) \geq n-2$. In particular, $\langle\sigma\rangle$ has only trivial representations of dimension less than $n-2$. Furthermore, any such representation is irreducible. $\diamondsuit$

*Proof of Proposition 4-4.3*

First note that $C_V(\tau) \neq \{0\}$ since $|\tau| = p$ and $V$ is defined over a field of characteristic $p$. Moreover, since $\sigma$ and $\tau$ commute, $C_V(\tau)$ is $\sigma$-invariant, and

$$[V, \sigma] := \{v - v\sigma \mid v \in V\}$$

is a non-trivial $\tau$-invariant subspace, hence $C_V(\tau) \cap [V, \sigma] \neq \{0\}$. Likewise, $C_V(\sigma)$ is $\tau$-invariant, hence $C_V(\tau) \cap C_V(\sigma) \neq \{0\}$.

Since $C_V(\tau)$ is $\sigma$-invariant and $(|\sigma|, p) = 1$, it follows from Maschke's theorem that $V = C_V(\tau) \oplus W$, for some $\sigma$-invariant subspace $W$. By Lemma 4-4.4, either $\dim C_V(\tau) \geq n-2$ or $\dim W \geq n-2$.

If $\dim C_V(\tau) \leq 2$, then $\sigma|_{C_V(\tau)} = e$, by our assumption on $|\sigma|$ and so $[V, \sigma] \subset W$. But then $\tau$ leaves $[V, \sigma]$ invariant, hence $C_V(\tau) \cap W \neq \{0\}$ which contradicts $V = C_V(\tau) \oplus W$. Otherwise, $\dim C_V(\tau) = n-2$ and $\dim W = 2$. In this case, $\sigma|_W = e$ and $\langle \sigma \rangle$ acts irreducibly on $C_V(\tau)$ by Lemma 4-4.4. Hence $W = C_V(\sigma)$. But $C_V(\sigma)$ is $\tau$-invariant, which implies that $C_V(\tau) \cap C_V(\sigma) \neq \{0\}$ contradicting $V = C_V(\tau) \oplus W$.

The only remaining possibility is $\dim C_V(\tau) = n-1$, and $\tau$ is a transvection. Moreover, $\dim C_V(\tau) \cap C_V(\sigma) = 1$ and $\sigma$ acts irreducibly on $[V, \sigma] \cap C_V(\tau)$, a hyper-hyperplane in $V$. $\diamondsuit$

Proposition 4-4.3 will be applied in the form of the following corollary which is easily derived.

**Corollary 4-4.5** *Let $x \in SL(n,q)$, $|x| = p \cdot p' \cdot r$, where $p$ is the characteristic of $GF(q)$, $p'$ is a primitive prime divisor of $q^{n-2}-1$ for $(q, n-2)$, and $r := \frac{|x|}{p\,p'}$. Then $\tau = x^{rp'}$ is a transvection, and $\sigma = x^{pr}$ commutes with $\tau$ and acts irreducibly on a hyperplane of $C_V(\tau)$.*

Corollary 4-4.5 has an obvious extension to $PSL(n,q)$.

**Corollary 4-4.6** *Let $\bar{x} \in PSL(n,q)$ satisfy $|\bar{x}| = p \cdot p' \cdot r$, where $p$ is the characteristic of $GF(q)$, $p'$ is a primitive prime divisor of $q^{n-2}-1$ for $(q, n-2)$, and $r := \frac{|x|}{p\,p'}$. Then $\bar{\tau} = \bar{x}^{rp'}$ is a transvection of $PSL(n,q)$, and $\bar{\sigma} = \bar{x}^{pr}$ is a ppd$(n,q,n-2)$ element. If $\tau'$ is a transvection in $\bar{\tau}$, then there exists a ppd$(n,q,n-2)$ element $\sigma'$ of $\bar{\sigma}$ such that $\sigma' \in SL(n,q)$ and $\sigma'$ commutes with $\tau'$ and acts irreducibly on a hyperplane of $C_V(\tau')$.*

*Proof:* Since $|\bar{\tau}| = p$ and $|\mathcal{Z}(SL(n,q))| = q-1$, $\bar{\tau}$ contains a unique element $\tau'$ of order $p$. This implies that $\tau'$ is central in the pre-image of $\langle \bar{x} \rangle$. In particular, $\tau'$ commutes with an element

$\sigma'$ which has order $p'$ where $p'$ is a primitive prime divisor of $q^{n-2}-1$ for $(q, n-2)$, hence $\sigma'$ is a *ppd*$(n, q, n-2)$ element of $SL(n, q)$ which commutes with $\tau'$. $\diamondsuit$

We now present a probabilistic argument which together with Corollaries 4-4.5 and 4-4.6 provide the basis for a Las Vegas algorithm for finding a transvection in a black box group isomorphic to $PSL(n, q)$ or $SL(n, q)$.

### 4-4.1 Probability argument

Let us estimate the probability that a randomly chosen element $x$ of $SL(n, q)$ or $PSL(n, q)$ will have order divisible by both $p$ and a primitive prime divisor $p'$ of $q^{n-2}-1$ for $(q, n-2)$.

Following Neumann and Praeger in [NeP], we call an element $y \in GL(n, q)$ *nearly irreducible* if $V$ may be expressed as a direct sum $V_1 \oplus V_2$ of $y$-invariant subspaces such that $V_1$ is one-dimensional and $y$ acts irreducibly on $V_2$. If the order of a nearly irreducible $y$ is divisible by a primitive prime divisor of $q^{n-1}-1$, $y$ is called *primitive*. Neumann and Praeger prove that for $n \geq 3$ the proportion of primitive nearly irreducible elements in $SL(n, q)$ or $GL(n, q)$ is at least $1/n$.

From Corollary 4-4.5 we know that if $x$ has the order $pp'r$ then there is a transvection $\tau = x^{p'r}$ and an element $\sigma$ of order $p'$ acting nearly irreducibly on $C_V(\tau)$. Thus, we need to establish that there is a reasonable probability of finding such an element $x$.

**Lemma 4-4.7** *Given a subgroup $G$, where $SL(n, q) \leq G \leq GL(n, q)$, and $n \geq 3$, the probability of finding an element $x$ of $GL(n, q)$ whose order is divisible by $pp'$ where $p$ is the characteristic of $GF(q)$ and $p'$ is a primitive prime divisor of $q^{n-2}-1$ for $(q, n-2)$ is at least $1/nq$. The same holds for $G$ satisfying $PSL(n, q) \leq G \leq PGL(n, q)$.*

*Proof:* Let $W$ be one of the $\frac{q^n-1}{q-1}$ hyperplanes in $V$ and consider the subgroup $A$ of $GL(V)$ which stabilizes $W$ and acts trivially on $V/W$. $A$ is isomorphic to a subgroup of the affine group on $W$ with kernel $K$ whose non-trivial elements consist of the $q^{n-1}-1$ transvections $\tau$

that fix $W$ and which is isomorphic to $V(n-1,q)$. $A$ contains a complement $H$ which acts as $GL(K)$ on $K$ with kernel a subgroup of $Z(GL(V))$. Each element of $H$ which acts as a nearly primitive irreducible element on $W$ centralizes a unique transvection subgroup of $K$ of order $q$ and the product of this element and any non-trivial element of the transvection subgroup will have order divisible by $pp'$, $p'$ a primitive prime divisor of $q^{n-2}-1$ for $(q,n-2)$.

If $d \geq 2$, then it is easy to show that the proportion of elements of $GL(d,q)$ which are primitive nearly irreducible is at least $1/d$. This follows from a result of [NeP] for $n \geq 3$ and is easy to prove directly when $d=2$. Applying this to $H$, it follows that each transvection of $K$ will be centralized by at least $\frac{|GL(n-1,q)|}{n-1}$ nearly primitive irreducible elements of $H$. Multiplying this estimate by the number of distinct hyperplanes in $V$, shows that the number of elements of $G$ of order divisible by $pp'$ for some $p'$ a primitive prime divisor of $q^{n-2}-1$ for $(q,n-2)$ is at least

$$\frac{q^n-1}{q-1}\cdot\frac{|GL(n-1,q)|}{n-1}\cdot(q^{n-1}-1).$$

Hence the probability that a random element $x \in G$ has order divisible by $pp'$ is at least

$$\frac{q^{n-1}-1}{(q-1)(n-1)q^{n-1}} > \frac{1}{nq} - \frac{1}{nq^{n-2}} > \frac{1}{2nq}.$$

Asymptotically, the latter probability is $\sim \frac{1}{nq}$. $\diamondsuit$

To proceed, we need an efficient algorithm for testing if a given element $x \in G$ has order divisible by $p$ and by some primitive prime divisor $p'$ of $q^{n-2}-1$ for $(q,n-2)$.

The first step is to test if $p$ divides $|x|$ and $|x|$ divides $p(q^{n-2}-1)$. This can be done by testing if

$$x^{q^{n-2}} \neq x \text{ and } x^{pq^{n-2}} = x^p.$$

To see why this condition works, note that $x^{pq^{n-2}} = x^p$ implies that $|x| \mid p(q^{n-2}-1)$ and $x^{q^{n-2}} \neq x$ implies that $p \mid |x|$. Using a simple doubling algorithm, it is easy to see that testing this condition is dominated by the time to perform $O(n \log q)$ multiplies.

Assuming that $p \mid |x|$ and $|x| \mid p(q^{n-2}-1)$, set $x' = x^p$. Then $|x'| \mid (q^{n-2}-1)$ and it remains to test if $|x'|$ is divisible by some primitive prime divisor $p'$ of $q^{n-2}-1$ for $(q, n-2)$. We use the strategy described by Neumann and Praeger in [NeP]. Set $d = n-2$. By means of a simple algorithm (see [NeP], sec. 6) the number $q^d - 1$ can be factored as

$$q^d - 1 = \Phi(d,q) \cdot \Psi(d,q)$$

where $\Phi(d,q)$ is the product of all primitive prime divisors of $q^d-1$ with multiplicities, and $\Psi(d,q)$ contains no primitive prime divisors. This one-time integer computation takes time $O(d^3 \log d (\log q)^4)$. After it is completed, testing whether $x' \in G$ is divisible by a primitive prime divisor of $q^d-1$ is equivalent to verifying that ${x'}^{\Psi(d,q)}$ is not the identity: This in turn can be achieved using $O(n \log q)$ multiplications, again using a simple doubling algorithm.

By Lemma 4-4, this test will have to be repeated for $O(nq)$ elements of $G$ in order to find an element $x$ whose order is divisible by $pp'$ where $p'$ is a primitive prime divisor of $q^{n-2}-1$ for $(q, n-2)$. Computing the total time taken for these tests, together with the one time cost to compute the factorization $q^d - 1 = \Phi(d,q) \cdot \Psi(d,q)$ then leads to the following result.

**Lemma 4-4.8** *There exists a Las Vegas algorithm which with constant reliability will produce an element of $x \in G$ of order $pp'$ where $p$ is the characteristic of $GF(q)$, and $p'$ is a primitive prime divisor of $q^{n-2}-1$ for $(q, n-2)$, in time*

$$O(n^3 \log n (\log q)^4 + nq[\rho + \mu n \log q]).$$

Notice that the elements $t$ and $s$ obtained as

$$t = x^{p'\,r}, \qquad s = x^{p\,r}$$

from an element $x$ of the above lemma, are *guaranteed* to be a transvection and a *ppd*$(q, n-2)$ element, provided that the values of $n, q$ used in the algorithm are the correct ones for $G$. Otherwise, the random trials will, with high probability, fail to produce the elements with the desired properties.

## 4-5 Computing with transvections in a black box group

In this section we describe a number of techniques that make it possible to study the action of $G$ on its set of transvections. Each time we will rely on identities involving transvections in $SL(n,q)$, while computing with the image transvections in $G$. Constructing simple "straight line programs" in $G$ will allow us to make statements about the transvections involved. The following lemma serves as an example.

**Lemma 4-5.1** *For two transvections $\tau_{v\otimes f}, \tau_{u\otimes g} \in SL(n,q)$ , the following cases are possible:*

**(i)** *$f(u) = g(v) = 0$, the transvections commute.*

**(ii)** *$f(u) \neq 0$ and $g(v) \neq 0$, the transvections do not commute and $\tau_{v\otimes f}^{\tau_{u\otimes g}}$ doesn't commute with $\tau_{v\otimes f}$.*

**(iii)** *$f(u) \neq 0$ and $g(v) = 0$, or $f(u) = 0$ and $g(v) \neq 0$, (i.e. $f(u)g(v) = 0$, but either $f(u) \neq 0$ or $g(v) \neq 0$), the transvections do not commute, and $\tau_{v\otimes f}^{\tau_{u\otimes g}}$ commutes with $\tau_{v\otimes f}$ but not with $\tau_{u\otimes g}$.*

*The asymptotic probabilities of these cases for random independently chosen transvections $\tau_{v\otimes f}$ and $\tau_{u\otimes g}$ are respectively $\frac{1}{q^2}$ and $(1-\frac{1}{q})^2$ for cases (i) and (ii), and $\frac{2}{q}(1-\frac{1}{q})$ for the case (iii).*

The statement of the lemma holds for any perfect central extension $G$ of $PSL(n,q)$. For any two transvections $t_{v\otimes f}, t_{u\otimes g} \in G$ we can determine the relations between $v, u, f, g$ by constructing quite short straight line programs out of

$$t_{v\otimes f}, \quad t_{u\otimes g}, \quad t_{v\otimes f}^{t_{u\otimes g}}, \quad t_{u\otimes g}^{t_{v\otimes f}} \; .$$

Since in our notation

$$t_{v\otimes f} = \Theta(\tau_{v\otimes f}), \quad t_{u\otimes g} = \Theta(\tau_{u\otimes g}), \quad t_{v\otimes f}^{t_{u\otimes g}} = \Theta(\tau_{v\otimes f}^{\tau_{u\otimes g}}), \quad t_{u\otimes g}^{t_{v\otimes f}} = \Theta(\tau_{u\otimes g}^{\tau_{v\otimes f}}) \; ,$$

identifying one of the three cases above leads us to a conclusion about $v, u, f, g$.

Notice that we cannot distinguish between the two subcases of case (iii) by using commutation relations between two transvections alone. Indeed, in that case $t_{u\otimes g}$ lies in one of the two maximal blocks of imprimitivity for $t_{v\otimes f}$, either $\Gamma(f)$ or $\beta(v)$, but since these blocks can be exchanged by an automorphism, nothing in the group structure can help us to distinguish them when only two transvections are known.

### 4-5.1 Key algorithms for transvections.

We now state key identities for transvections in $SL(n,q)$ that allow us to construct the homomorphism $\Theta$ by labeling the elements of the black box group $G$ consistently. They follow from the identity

$$(\tau_{v\otimes f}\cdot\tau_{u\otimes g})(w) = w + f(w)(v+g(v)\,u) + g(w)\,u \qquad \forall w\in V\ .$$

It should be noted that the product of two transvections need not be a transvection in general, as the above identity shows.

$$\tau_{v_1\otimes f}\cdot\tau_{v_2\otimes f} = \tau_{(v_1+v_2)\otimes f}, \tag{4.1}$$

$$\tau^{\xi}_{v\otimes f} = \tau_{\xi(v\otimes f)} = \tau_{v\xi\otimes\xi^* f}, \quad \xi\in SL(n,q), \tag{4.2}$$

$$\tau^{c}_{v\otimes f} = \tau_{cv\otimes f} = \tau_{v\otimes cf}, \quad \forall c\in\mathbb{F}_p \tag{4.3}$$

$$(\tau_{v\otimes f})^{-1} = \tau_{-v\otimes f} \tag{4.4}$$

$$\tau^{\tau_{u\otimes g}}_{v\otimes f} = \tau_{(v-g(v)u)\otimes(f+f(u)g)}, \tag{4.5}$$

In equation (4.3), by $\tau^c_{v\otimes f}$ we mean $\tau_{v\otimes f}$ taken to the corresponding *integer* power, whereas on the right hand side of (4.3) $c$ is treated as an element of the finite field, i.e. a scalar for our linear spaces. Clearly, equation (4.3) immediately follows from (4.1). By choosing this notation, we are silently identifying $1,2,\ldots,p-1$ in $\mathbb{Z}$ and $\mathbb{F}_p$. This notation has a special meaning for our recognition purposes – we emphasize that the cyclic group formed by the powers of a single transvection is isomorphic to $\mathbb{F}_p^+$.

Applying these to constructing $\Theta$, we see that for transvections $t_{v_1\otimes f}, t_{v_2\otimes f} \in G$ such that $\Theta(\tau_{v_1\otimes f}) = t_{v_1\otimes f}$, $\Theta(\tau_{v_2\otimes f}) = t_{v_2\otimes f}$ we know that

$$t_{v_1\otimes f} \cdot t_{v_2\otimes f} = \Theta(\tau_{v_1\otimes f} \cdot \tau_{v_2\otimes f})$$

hence the notation $t_{(v_1+v_2)\otimes f}$ for the transvection obtained as $t_{v_1\otimes f} \cdot t_{v_2\otimes f}$ is consistent with respect to $\Theta$. Other identities are treated analogously.

It is worth pointing out that the right hand side of equation (4.5) doesn't change if we take $\tau_{\lambda v\otimes\lambda^{-1}f}$, $\tau_{\mu u\otimes\mu^{-1}g}$ instead of $\tau_{v\otimes f}$, $\tau_{u\otimes g}$, i.e. our notation is indeed convenient.

Notice that the identity (4.3) gives us a convenient way of representing all $\mathbb{F}_p$-multiples of $v \otimes f \in V \otimes V^*$. Namely, given a transvection $t_{v\otimes f}$, we can obtain all transvections $t_{cv\otimes f}$, $c = 2, \dots, p-1 \in \mathbb{F}_p$ by performing $p-1$ black box multiplications.

In Section 4-6 we do the same for multiplication by units in $GF(q)$, by showing how to construct all elements of the *full transvection group* $T_{v\otimes f} \subset G$, where

$$T_{v\otimes f} = \Theta(\{\tau_{\alpha v\otimes f} \mid \alpha \in GF(q)\}) = \Theta(\{\tau_{v\otimes\alpha f} \mid \alpha \in GF(q)\}) \cong GF(q)^+ ,$$

and how to use it as a model of the field $GF(q)$ in computations.

Throughout this section, we speak of transvections denoted by $\tau_{v\otimes f}$, $\tau_{u\otimes g}$, etc. The specific values of $u, v, f, g, \dots$ are not known (and actually only defined up to a constant for a pair $v \otimes f$ defining a transvection), while the output of each algorithm is defined in terms of the input's $u, v, f, g$. In other words, these algorithms guarantee that the labels on transvections they output are coherent with those of the inputs and $\Theta$-consistent.

**Algorithm** `IsProp` – **proportionality check.**

This algorithm takes as input two transvections, $t_{v\otimes f}$ and $t_{u\otimes g}$, and, for a non-prime $q$, the full transvection group of one of them, say, $T_{v\otimes f}$, decides whether these transvections have the same axis and proportional centers, i.e. whether $\langle v\rangle = \langle u\rangle$ and $\langle f\rangle = \langle g\rangle$, or, in our vector–functional pair notation, $\lambda v \otimes f = u \otimes g$ for some $\lambda \in GF(q)$. If this is the case, the algorithm computes and returns that $\lambda$, using the model of the field $GF(q)$. The pseudo–code for this function will be given in Sec. 4-6.1, after we construct our model of the field $GF(q)$.

Suppose that we have a way of constructing the full transvection group $T_{v\otimes f} = \{t_{\alpha v\otimes f} : \alpha \in GF(q)\}$ of the first argument $t_{v\otimes f}$. By exhaustive search over the set $T_{v\otimes f}$ we can check whether $t_{u\otimes g} \in T_{v\otimes f}$, and, if so, return $\lambda \in GF(q)$ such that $t_{\lambda v\otimes f} = t_{u\otimes g}$, i.e. $\lambda v \otimes f = u \otimes g$. We may not want to store the entire full transvection group $T_{v\otimes f}$, but rather generate all of its elements when necessary from a system of generators $\{t_{\theta^k v\otimes f} : k = 0, \ldots, l-1\}$.

Let, as above, $\mu$ be the time required for a black box multiplication, $\rho$ the time required to produce a (nearly) uniformly distributed random element of $G$. We assume that the bit string representing an element of $G$ is unique, therefore comparisons of elements are fast and do not affect complexity.

`IsProp` : [Check if two transvections are proportional over $\mathbb{F}_p$. ]

*Input:* $(t_{v\otimes f}, t_{u\otimes g}, T_{v\otimes f})$, two transvections, and a full transvection group.

*Output:* $\lambda \in GF(q)$ such that $t_{\lambda v\otimes f} = t_{u\otimes g}$ or `false` if no such $\lambda$ exists.

*Complexity:* $O(q\mu)$.

See Section 4-6.1.

**Algorithm** `IsZero`**.**

For two transvections $t_{v\otimes f}$ and $t_{u\otimes g}$ `IsZero` determines whether $f(u) = 0$ and/or $g(v) = 0$. More exactly, it distinguished between the three cases of Lemma 4-5.1, (i) $f(u) = g(v) = 0$, (ii) $f(u) \neq 0,\ g(v) \neq 0$, (iii) $f(u) = 0,\ g(v) \neq 0$ or $f(u) \neq 0,\ g(v) = 0$.

This information is provided by commutation relations between $t_{v\otimes f}$, $t_{u\otimes g}$ and $t_{v\otimes f}^{t_{u\otimes g}}$ in Lemma 4-5.1.

`IsZero` : [Check if functionals annihilate vectors realized by transvections.]

*Input:* $t_1 = t_{v\otimes f}, t_2 = t_{u\otimes g}$.

*Output:* One of "Case 1", "Case 2" , "Case 3".

**Case 1:** $f(u) = g(v) = 0$.

**Case 2:** $f(u) \neq 0,\ g(v) \neq 0$.

**Case 3:** $f(u) \neq 0,\ g(v) = 0$ or $f(u) = 0,\ g(v) \neq 0$.

*Complexity:* $O(\mu)$.

`If` $t_1\, t_2 = t_2\, t_1$ `Return`("Case 1").

`If` $t_1^{t_2}\, t_1 \neq t_1\, t_1^{t_2}$ `Return`("Case 2").

`If` $t_1^{t_2}\, t_1 = t_1\, t_1^{t_2}$ `and` $t_1^{t_2}\, t_2 \neq t_2\, t_1^{t_2}$ `Return`("Case 3").

### Algorithms `FindVect` and `FindFunc`

Given a transvection $t_{v\otimes f}$, construct a transvection of the form $t_{u\otimes f}$, where $\langle u\rangle \neq \langle v\rangle$ ( `FindVect` ) and a transvection of the form $t_{v\otimes g}$, where $\langle g\rangle \neq \langle f\rangle$ ( `FindFunc` ). Geometrically, this means constructing, for a transvection $t_{v\otimes f}$, a pair of transvections such that one of these has the same axis as $t_{v\otimes f}$ and a non-proportional center ( `FindVect`), and the other one has a proportional center but a different axis ( `FindFunc`). Equivalently, one of these two transvections lies in $\Gamma(f)$, and the other one lies in $\beta(v)$. As we have remarked in Lemma 4-5.1, we cannot distinguish between these two blocks having only $t_{v\otimes f}$ and another transvection from one of the blocks. However, once we obtain the first of the desired transvections and (arbitrarily) assign it a label $t_{u\otimes f} \in \Gamma(f)$, we make sure that the other transvection we are going to construct belongs to the other block $\beta(v)$.

Observe that in $SL(n,q)$ the commutator

$$[\tau_{v\otimes f}, \tau_{u\otimes g}] = \tau_{v\otimes f}^{-1}\tau_{u\otimes g}^{-1}\tau_{v\otimes f}\tau_{u\otimes g} = \tau_{v\otimes f}^{-1}\tau_{v\otimes f}^{\tau_{u\otimes g}}$$

of $\tau_{v\otimes f}$, $\tau_{u\otimes g}$ behaves as follows:

| | $f(u)=0$ | $f(u)\neq 0$ |
|---|---|---|
| $g(v)=0$ | $id$ | $\tau_{f(u)v\otimes g}$ |
| $g(v)\neq 0$ | $\tau_{-g(v)u\otimes f}$ | $\notin \mathcal{T}$ |

In particular, for two transvections $\tau_{u\otimes f} \in \Gamma(f)$, $\tau_{v\otimes g} \in \beta(v)$ we have $[\tau_{u\otimes f}, \tau_{v\otimes g}] = \tau_{g(u)v\otimes f}$, i.e. the commutator is, with high probability, a non-trivial transvection. On the contrary, for two transvections $\tau_{u\otimes f}, \tau_{w\otimes f} \in \Gamma(f)$ the commutator is the identity, and analogously for any two transvections from the block $\beta(v)$.

For $\Theta$ to be a homomorphism, analogous relations should hold in $G$ for two transvections $t_{v\otimes f}$, $t_{u\otimes g}$. We can distinguish between the three cases (which correspond to all possible relative orientations of the centers and axes of the two transvections) by using the algorithm `IsZero`. For example, if `IsZero` determines that for some $t_{v\otimes f}$, $t_{u\otimes g}$ such that $t_{v\otimes f} = \Theta(\tau_{v\otimes f})$, $t_{u\otimes g} = \Theta(\tau_{u\otimes g})$ the "Case 3" of `IsZero` takes place, then

$$[t_{v\otimes f}, t_{u\otimes g}] = \Theta([\tau_{v\otimes f}, \tau_{u\otimes g}]) = \Theta(\tau_{g(v)u\otimes f})$$

and hence $[t_{v\otimes f}, t_{u\otimes g}]$ can be consistently denoted either as $t_{-g(v)u\otimes f}$ or as $t_{f(u)v\otimes g}$. For the very first choice of the commutator result's label we can choose either label, while subsequent choices have to agree with the previous choices.

We can construct the first transvection in the block $\Gamma(f)$ by conjugating $t_{v\otimes f}$ with a random $x \in G$, to get some $t_{v'\otimes f'} = t_{v\otimes f}^x$, and checking whether $f'(v) \neq 0$, $f(v') = 0$ or $f'(v) = 0$,

$f(v') \neq 0$ (Case (iii) of Lemma 4-5.1) by `IsZero`. If so then denote

$$[t_{v'\otimes f'}, t_{v\otimes f}] = t_{f'(v)v'\otimes f} \ \in \Gamma(f) \ .$$

If $v' \neq \lambda v$ (which is checked by `IsProp`), we are done. We will refer to this procedure as `FindVect`.

The procedure for obtaining a $t_{v\otimes g}$ given $t_{v\otimes f}$ (with $g \neq \lambda f$, $\lambda \in GF(q)$) is analogous, but now we also check that the transvection found by random conjugation and the commutation with $t_{v\otimes f}$ indeed lies in the other maximal imprimitivity block $\beta(v)$, by checking if it commutes with the result of `FindVect` . We call this second procedure `FindFunc`. Both will succeed with constant probabilities, `FindVect` with probability $\sim \frac{1}{q}(1-\frac{1}{q})$, and `FindFunc` with probability $\sim \frac{1}{2q}(1-\frac{1}{q})$ by the Lemma 4-5.1.

These two procedures are Las Vegas. `FindVect` fails with probability of $\frac{q-1}{q^{n-1}-1} \sim \frac{1}{q^{n-2}}$, which is the probability that a random vector in $Ker(f)$ is a multiple of a fixed vector in the same hyperplane. `FindFunc` can take a longer time than `FindVect` , by finding a transvection from the same maximal block as the one provided earlier by `FindVect` , with probability $\frac{1}{2}$. Both procedures need to call `Random(G)` to produce a uniform random element of the black box group $G$; the data structures representing $G$ in the following pseudo–code are assumed to be a part of the global environment. More importantly, it is assumed that the global environment also contains such data structures representing the full transvection group $T_{v\otimes f}$ of the argument $t_{v\otimes f}$ that allow us to perform complete search over it. This is always true for the contexts in which `FindVect` and `FindFunc` are called, i.e. a system of generators for the full transvection group of the argument of either of these functions is always pre-computed. Since these data structures are only required for *checking* the result, not for obtaining it, and the probability of failure is low, we chose not to include them among the input parameters, but rather assume them as a part of the global environment.

`FindVect` : [Find a transvection with the same axis but a different center]

*Input:* $t_{v\otimes f}$, the original transvection.

*Output:* $t_{u\otimes f}$ such that $\langle u\rangle \neq \langle v\rangle$ or `fail`.

*Globals:* $G$, $T_{v\otimes f}$.

*Reliability:* With constant probability of success $> \frac{1}{q^{n-2}}$.

*Complexity:* $O(q[\rho+\mu]+q\mu) = O(q[\rho+\mu])$.

`Repeat`

  `Set` $x \leftarrow$ `Random`$(G)$

  `Set` $t \leftarrow t^x_{v\otimes f}$

`Until IsZero`$(t_{v\otimes f}, t) =$ "Case 3"

[To encounter Case 3, $O(q)$ tries are required, see 4-5.1.]

`If IsProp`$(t_{v\otimes f}, [t_{v\otimes f}, t], T_{v\otimes f}) \neq$ `false`

  `Return(fail)`

  [the result is a multiple of the given vector, throw it away.]

`Return`( $[t_{v\otimes f}, t]$ )

The pseudo–code for the procedure `FindFunc` is analogous, with an added check to to verify that the found transvections belongs to the other maximal imprimitivity block, designated $\beta(v)$:

`FindFunc` : [Find a transvection with the same center but a different axis]

*Input:* $(t_{v\otimes f},\, t_{u\otimes f})$, where $t_{u\otimes f}$ is a transvection found by `FindVect` .

*Output:* $t_{v\otimes h}$ such that $\langle h\rangle \neq \langle f\rangle$ or `fail`.

*Globals:* $G$, $T_{v\otimes f}$.

*Reliability:* With constant probability of success $> \frac{1}{q^{n-2}}$.

*Complexity:* $O(q[\rho+\mu]+\mu q) = O(q[\rho+\mu])$.

`Repeat`

  `Set` $x \leftarrow$ `Random`$(G)$

  `Set` $t \leftarrow t_{v\otimes f}^{x}$

`Until IsZero`$(t, t_{v\otimes f}) =$ "Case 3" `and` $[[t_{v\otimes f}, t], t_{u\otimes f}] \neq id$

`If IsProp`$(t_{v\otimes f}, [t_{v\otimes f}, t], T_{v\otimes f}) \neq$ `false` [check proportionality]

  `Return(fail)`

`Return(` $[t_{v\otimes f}, t]$ `)`

**Algorithm `FindConst` – the key to solving equations.**

Given two transvections $t_{v\otimes f}, t_{u\otimes g}$, and, for a non–prime $q$, the full transvection group of one of them, say, $T_{v\otimes f}$, find $f(u)\cdot g(v)$ explicitly in $GF(q)$.

Take $t_{v\otimes f}$, $t_{u\otimes g}$ such that $f(u)\neq 0$, $g(v)\neq 0$ (which can be checked by `IsZero`). By (4.5)

$$t_{v'\otimes f'} := t_{v\otimes f}^{t_{u\otimes g}} = t_{(v+g(v)u)\otimes(f-f(u)g)},$$

$$t_{u'\otimes g'} := t_{u\otimes g}^{t_{v\otimes f}} = t_{(u+f(u)v)\otimes(g-g(v)f)}, \tag{4.6}$$

and further we see

$$g'(v') = (g-g(v)f)(v+g(v)u) = g(v)(1-g(v)f(u)),$$

$$f'(u') = (f-f(u)g)(u+f(u)v) = f(u)(1-g(v)f(u)).$$

Consider $t_{v'\otimes f'}$ and $t_{u'\otimes g'}$. The algorithm `IsZero` allows us to check whether $f'(u')=0$ and $g'(v')=0$. Observe that

$$g'(v')=0 \text{ and } f'(u')=0 \Leftrightarrow g(v)f(u)=1.$$

Suppose now that for a given $c \in GF(q)$ and $t_{v\otimes f}$ we could construct the transvection $t_{cv\otimes f}$ (for $c \in \mathbb{F}_p$ this is trivial, see (4.3)). Then, by exhaustive search over all $c \in GF(q)$, we will eventually find $c_0$ such that $c_0 f(u)g(v) = 1$, i.e. such that $t_{u\otimes g}^{t_{cv\otimes f}}$ and $t_{cv\otimes f}^{t_{u\otimes g}}$ commute.

Thus, given any two transvections $t_{v\otimes f}, t_{u\otimes g}$ we can actually *explicitly* compute the scalar $f(u)g(v)$ at the cost of an exhaustive search over the elementary abelian group $T_{v\otimes f}$ representing the finite field!

We call this procedure `FindConst`. Since it depends on our ability to construct the full transvection group of a given transvection, we will postpone its pseudo–code (as well as that of `IsProp`) until after our construction of the model of the field is described.

## 4-6 Constructing the model of $GF(q)$

For finite fields $\mathbb{F}_p$ of prime sizes writing the missing procedures `IsProp` and `FindConst` is easy, because of (4.3). We recommend that the reader who just wants to understand the general idea of our method or is interested only in finite fields of prime size skips this section and goes directly to Section 4-7 and the construction of the dual pair.

We will, as before, denote the characteristic of $GF(q)$ by $p$, $q = p^l$, $l > 1$.

Given a transvection $t_{v\otimes f}$ we want to generate the entire elementary abelian subgroup

$$T_{v\otimes f} = \{t_{\alpha v\otimes f} : \alpha \in GF(q)\} \cong GF(q)^+$$

and use it, as a model of $GF(q)$, for exhaustive searches over the field and other computations with the elements of $GF(q)$.

**Theorem 4-6.1** *Given the original transvection $t_{v\otimes f} \in G$, the entire group $T_{v\otimes f}$ can be generated in $O(q[\mu + \rho])$ time, with constant probability of success. In the process $l$ generators*

$$t_{\theta^i v\otimes f}, \qquad \theta \in GF(q)^\times,\ i = 0, \dots, l-1$$

*are obtained, such that $1, \theta, \theta^2, \dots, \theta^{l-1}$ form an $\mathbb{F}_p$-basis of $GF(q)$.*

From this point on the elements of $GF(q)$ will be represented by $l$-tuples of their coordinates in the basis $1, \theta, \theta^2, \ldots, \theta^{l-1}$, i.e. we will be considering $GF(q)$ as the field of polynomials over $\mathbb{F}_p$ modulo the ideal generated by $f_\theta(x)$ in $GF(q)[x]$, where $f_\theta(x)$ is the minimal polynomial of $\theta$. To compute with such $l$-tuples we need to know $f_\theta(x)$. We don't need to express $\theta \in GF(q)$ explicitly, and it is, in fact, enough to find for our $t_{v\otimes f}$ any $t_{\theta v\otimes f}$, where $\theta \in GF(q)$ lies in no proper subfield of $GF(q)$.

We start with $t_{v\otimes f}$ and want to find two transvections of the forms $t_{u\otimes f}$ and $t_{v\otimes h}$ from the blocks $\Gamma(f)$ and $\beta(v)$ respectively, such that $\langle v\rangle \neq \langle u\rangle$ and $\langle h\rangle \neq \langle f\rangle$, i.e. one of these has the common axis with $t_{v\otimes f}$ and the other has a proportional center, but they respectively differ from $t_{v\otimes f}$ in the other component. They can be found in $O(q[\rho+\mu])$ time, with constant probability of success, as follows.

For randomly chosen $x, y \in G$ denote

$$t_{v'\otimes f'} := t^x_{v\otimes f}, \qquad t_{v''\otimes f''} := t^y_{v\otimes f}\,.$$

We know (Lemma 4-5.1) that for these two transvections the probability of "Case 3" is approximately $(1-1/q)2/q$. Without loss of generality, assume $f'(v) \neq 0$, $f(v') = 0$ (for this procedure either an element of $\Gamma(f)$ or $\beta(v)$ will do, as long as the other transvection will be from the other block. We will not use the labels of the auxiliary transvections for construction of $\Theta$). Trying independent random elements $x \in G$, we will need $O(q)$ tries before we locate, with constant probability of success, an $t_{v'\otimes f'} = t^x_{v\otimes f}$ for which this is the case. Analogously, finding $y \in G$ such that $t_{v''\otimes f''} = t^y_{v\otimes f}$ satisfies $f''(v) = 0$, $f(v'') \neq 0$, i.e. lies in the other maximal imprimitivity block, say, $\beta(v)$ should take no more than $O(q)$ random tries. Notice that procedures `FindVect` and `FindFunc` use similar reasoning.

The relations

$$f'(v)\cdot f(v') = 0, \qquad f''(v)\cdot f(v'') \neq 0 \qquad [[t_{v''\otimes f''}, t_{v\otimes f}], [t_{v'\otimes f'}, t_{v\otimes f}]] \neq id.$$

are verified by `IsZero` and by straightforward black box multiplications and comparisons. If they do not hold, we try other random $x$ or $y$.

If this is the case, then recall from 4-5.1 that if the above relations hold, then this implies

$$[t^x_{v\otimes f}, t_{v\otimes f}] = t_{f'(v)v'\otimes f} \quad \text{and} \quad [t^y_{v\otimes f}, t_{v\otimes f}] = t_{v\otimes f(v'')f''}.$$

Therefore we have the desired two transvections that share with $t_{v\otimes f}$ the axis and the center respectively, and belong to different maximal imprimitivity blocks for $t_{v\otimes f}$. To emphasize this fact, denote

$$\begin{aligned} t_{u\otimes f} &:= t_{f'(v)v'\otimes f} = [t^x_{v\otimes f}, t_{v\otimes f}], \\ t_{v\otimes h} &:= t_{v\otimes f(v'')f''} = [t^y_{v\otimes f}, t_{v\otimes f}]. \end{aligned} \tag{4.7}$$

Further, let $v_1 = v, v_2, \ldots, v_n$ be a basis of $V$ with the corresponding dual basis $g_1, \ldots, g_{n-1}, g_n = f$ in $V^*$, $g_i(v_j) = \delta_{ij}$. Then we can write

$$t_{u\otimes f} = t_{(a_1v_1+\ldots+a_{n-1}v_{n-1})\otimes f}, \qquad t_{v\otimes h} = t_{v\otimes(b_2g_2+\ldots b_ng_n)},$$

and

$$[t_{v\otimes h}, t_{u\otimes f}] = t_{h(u)v\otimes f} = t_{(a_2b_2+\ldots+a_{n-1}b_{n-1})v\otimes f}.$$

Unless $h(u) = 0$ (in which case the commutator is equal to $id$), we get a new transvection $t_{\alpha v\otimes f}$ from $T_{v\otimes f}$ $(\alpha = a_2b_2 + \ldots + a_{n-1}b_{n-1})$. For block matrices in a suitable basis this reads

$$\left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & I & 0 \\ a & u & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ h & I & 0 \\ b & 0 & 1 \end{pmatrix}\right] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & I & 0 \\ h(u) & 0 & 1 \end{pmatrix}.$$

For each new $t_{v\otimes h}$ we get another element of $T_{v\otimes f}$, at the cost $O(q[\rho + \mu])$. Moreover, if $a_2, \ldots, a_{n-1}, b_2, \ldots, b_{n-1}$ are the coordinates of a uniformly distributed non-zero vector $u \in V$ and a uniformly distributed non-zero functional $h \in V^*$, then $h(u) = a_2b_2 + \ldots + a_{n-1}b_{n-1}$ takes any value from $GF(q)^\times$ with equal probability. Indeed, for a fixed non-zero functional $h \in V^*$ and a random uniformly distributed non-zero vector $w \in V$, the value of $h(w)$ is uniformly distributed, as long as the pairing is non-degenerate. On condition that $h(u)$ above is non-zero, i.e.

$$[t_{v\otimes h}, t_{u\otimes f}] \neq e,$$

neither $h$ nor $u$ can be zero, hence their scalar product $h(u)$ is uniformly distributed. Thus we have a way of producing a uniformly distributed random element in the transvection group

$$T_{v\otimes f} = \{t_{\alpha v\otimes f} \mid \alpha \in GF(q)\} \cong GF(q)^+,$$

given one of its elements $t_{v\otimes f}$.

To compute with units of $GF(q)$ we need a set of generators of the form

$$\{t_{v\otimes f}, t_{\theta v\otimes f}, t_{\theta^2 v\otimes f}, \ldots, t_{\theta^{l-1} v\otimes f}\},$$

where $\theta \in GF(q)$ lies in no proper subfield of $GF(q)$. We can obtain this set as follows.

Given $t = t_{v\otimes f}$, use the procedure described above to get a $t' = t_{\theta v\otimes f}$, where $\theta$ is a random element of $GF(q)$. With probability greater than $1 - 1/p^2$ such a random element lies is no proper subfield of $GF(q)$. Recall that in (4.7) we denoted

$$t_{u\otimes f} := [t_{x(v\otimes f)}, t_{v\otimes f}] = [t^x_{v\otimes f}, t_{v\otimes f}] = [t^x, t],$$

$$t_{v\otimes h} := [t_{y(v\otimes f)}, t_{v\otimes f}] = [t^y_{v\otimes f}, t_{v\otimes f}] = [t^y, t],$$

and

$$t' := t_{\theta v\otimes f} = t_{h(u) v\otimes f} = [t_{v\otimes h}, t_{v\otimes f}],$$

where $x$, $y$ are suitable random elements of $G$, i.e $x$, $y$ satisfy the relations 4.7 above. For transvections in $SL(n,q)$, $\xi \in SL(n,q)$ we have, according to 4-5.1 :

$$[(t')^x, t] = [t^x_{\theta v\otimes f}, t_{v\otimes f}] = [t_{x(\theta v\otimes f)}, t_{v\otimes f}] = [t_{\theta\, x(v\otimes f)}, t_{v\otimes f}] = t_{\theta u\otimes f},$$

where we use the fact that the action of $x \in G$ on $V \otimes V^*$ is linear, and last equality follows from the definition of $\tau_{u\otimes f}$ above. Further,

$$[t_{v\otimes h}, t_{\theta u\otimes f}] = t_{\theta^2 v\otimes f},$$

and, analogously,

$$t_{\theta^3 v\otimes f} = [t_{v\otimes h},\, [t^x_{\theta^2 v\otimes f}, t_{v\otimes f}]\,] \quad \text{etc.}$$

Therefore we can assign the labels

$$t_{\theta u\otimes f} := [(t')^x, t], \quad t_{\theta^2 v\otimes f} := [t, [(t')^x, t]] \quad \text{etc.}$$

Computing the minimal polynomial $f_\theta(t)$ of $\theta$ takes $O(q\mu)$ time, by exhaustive search. Indeed, denoting $T(\alpha) = \tau_{\alpha v\otimes f}$ we have

$$\prod_{k=0}^{l-1} T(\theta^k)^{a_k} = T(\sum_{k=0}^{l-1} a_k\theta^k),$$

where $(a_0, a_1, \ldots, a_{l-1}) \in \mathbb{F}_p^l$. Using Horner's rule to construct the products, we will need $O(q)$ multiplications and comparisons.

In the process it will be verified that $\theta$ indeed lies in no proper proper subfield of $GF(q)$ — in that case $\deg f_\theta(t) = l$. If we discover that $\deg f_\theta(t) < l$, we have to discard the random elements $x, y$ together with the newly constructed transvections, and start over. This procedure is Las Vegas.

Notice that we need to find only two random elements for the entire procedure (while we still need $O(q)$ tries to locate a suitable random element). Therefore, the complexity is reduced to $O(q[\rho+\mu] + l\mu + q\mu) = O(q[\rho+\mu])$, with the probability of success greater than $1 - 1/p^2$ (more exactly, $> 1 - 1/p^d$, where $d$ is the smallest prime divisor of $l$).

The pseudo–code for constructing a basis of the full transvection group $T_{v\otimes f}$ given a transvection $t_{v\otimes f}$ can be written as follows.

`Build-transvection-basis` : [Construct a basis for $T_{v\otimes f}$ ]

*Input:* $(t_{v\otimes f}, G)$

*Output:* $\{t_{\theta^k v\otimes f} \,:\, k = 0, \ldots, l-1\}$ for some $\theta \in GF(q)$, $\theta$ not in any proper subfield of $GF(q)$.

*Complexity:* $O(q[\mu+\rho])$.

`Set` $t_0 \leftarrow t_{v\otimes f}$ [ Denote $t_0 = t_{v\otimes f} = t_{\theta^0 v\otimes f}$ ]

`Repeat`

  `Set` $x \leftarrow$ `Random`$(G)$

  `Set` $t \leftarrow t_0^x$

`Until IsZero`$(t_0, t) =$ "Case 3"

[To encounter Case 3, $O(q)$ tries are required, see 4-5.1.]

`Set` $t_a \leftarrow [t_0, t]$ [ Obtained $t_{u\otimes f}$ ]

`Repeat`

  Set $y \leftarrow$ `Random`$(G)$

  `Set` $t \leftarrow t_0^y$

`Until IsZero`$(t, t_{v\otimes f}) =$ "Case 3" `and` $[[t_0, t], t_a] \neq id$

`Set` $t_b \leftarrow [t_0, t]$ [ Obtained $t_{v\otimes h}$ ]

`If` $[t_b, t_a] = id$ `Return( fail )` [ Repeat from the start ]

`Set` $t_1 \leftarrow [t_b, t_a]$ [ Obtained $t_{\theta v\otimes f}$ ]

`Set` $t_2 = [t_b, [t_1^x, t_0]]$ [ Obtained $t_{\theta^2 v\otimes f}$ ]

`For` $i \leftarrow 3$ `to` $l-1$

  `Set` $t_i \leftarrow [t_b, [t_{i-1}^x, t_0]]$ [ Obtained $t_{\theta^i v\otimes f}$ ]

[Find the coefficients of all candidates for the minimal polynomial of $\theta$ ]

`Set` $S \leftarrow \{\ (a_0, \ldots, a_{l-1}) \in (\mathbb{F}_p)^l \ \mid\ \prod_{i=0}^{l-1} t_i^{a_i} = id\ \}$

`If` (minimal degree of polynomials in $S$) $< l-1$ `Return( fail )`

`Return(` $\{t_0, t_1, \ldots, t_{l-1}\}$ `)`

### 4-6.1 Computing with scalars from $GF(q)$.

In the previous section we have shown how, starting with an arbitrary transvection $t = t_{v\otimes f}$ we can construct the full transvection group

$$T_{v\otimes f} = \{t_{\alpha v\otimes f} \mid \alpha \in GF(q)\} \cong GF(q)^+$$

and find its generators of the form $t_{\theta^k v\otimes f}$, $k = 0, \ldots, l-1$. Here $\theta$ may be any element of $GF(q)$ not in any proper subfield of $GF(q)$.

Since $1, \theta, \ldots, \theta^{l-1}$ form an $\mathbb{F}_p$-basis of $GF(q)$, any element of $T_{v\otimes f}$ can be uniquely represented as a product

$$t_{\alpha v\otimes f} = \prod_{k=0}^{l-1} (t_{\theta^k v\otimes f})^{a_k},$$

where $\alpha = a_0 + a_1\theta + \ldots + a_{l-1}\theta^{l-1}$, $a_k \in \mathbb{F}_p$. We are now ready to write the pseudo–code for `FindConst`.

Recall that, given two transvections $t_{v\otimes f}$ and $t_{u\otimes g}$, the commutation relations (4.6) and Lemma 4-5.1 allow us to check if $f(u)g(v) = 1$. Exhaustive search over $\{t_{\alpha v\otimes f}, \alpha \in GF(q)\}$ yields a unique $\alpha \in GF(q)$ such that $g(v)f(u) = \alpha^{-1}$. We can check if two transvections are proportional in the same fashion.

`FindConst` : [Find constants in $GF(q)$ explicitly.]

*Input:* $(t_{v\otimes f}, t_{u\otimes g}, \{t_{\theta^k v\otimes f}, k = 0, \ldots, l-1\})$

*Output:* $\mathbf{b} = (b_0, \ldots, b_{l-1}) \in \mathbb{F}_p \times \ldots \times \mathbb{F}_p$ such that

$$b_0 + b_1\theta + \ldots + b_{l-1}\theta^{l-1} = f(u)g(v)$$

*Complexity:* $O(q\mu)$ (see below).

`If IsZero`$(t_{v\otimes f}, t_{u\otimes g}) \neq$ "Case 2" `Return`$(\mathbf{0})$

`For` $\mathbf{b} = (b_0, \ldots, b_{l-1}) \in (\mathbb{F}_p)^l = \mathbb{F}_p \times \ldots \times \mathbb{F}_p$

Set $t \leftarrow \prod_{k=0}^{l-1} t_{\theta^k v \otimes f}^{b_k}$

If IsZero$(t_{u\otimes g}^{t}, t^{t_{u\otimes g}})$ = "Case 1" Return$(\mathbf{b}^{-1})$

[return inverse for further convenience]

The product $t = \prod_{k=0}^{l-1} t_{\theta^k v \otimes f}^{b_k}$ should actually be formed by Horner's rule, so that the cycle over $\{(b_0, \ldots, b_{l-1}) \mid b_i \in \mathbb{F}_p\}$ requires only one multiplication for each $l$-tuple, and the overall complexity is indeed $O(q\mu)$ as indicated.

IsProp : [Check if two transvections are proportional over $GF(q)$. ]

*Input:* $(t_{v\otimes f}, t_{u\otimes g}, \{t_{\theta^k v\otimes f},\ k = 0, \ldots, l-1\})$

*Output:* $\mathbf{b} = (b_0, \ldots, b_{l-1}) \in GF(q)$ such that $t_{u\otimes g} = t_{(\sum_{k=0}^{l-1} b_k \theta^k) v \otimes f}$ or false if no such $\mathbf{b}$ exists.

*Complexity:* $O(q\mu)$.

For $\mathbf{b} = (b_0, \ldots, b_{l-1}) \in (\mathbb{F}_p)^l = \mathbb{F}_p \times \ldots \times \mathbb{F}_p$

If $\prod_{k=0}^{l-1} t_{\theta^k v \otimes f}^{b_k} = t_{u\otimes f}$ Return$(\mathbf{b})$.

Return(false) [no such $\mathbf{b}$ found].

Notice that none of this is necessary when $q$ is itself prime. Simple checking of the first $p$ powers of one transvection would suffice.

It would seem that whenever we need to find $f(u)g(v)$ for some pair $t_{v\otimes f}, t_{u\otimes g}$ we would have to construct the full transvection group of one of them (getting a different $\theta$ each time!). In fact, we need to carry out this construction only once, for the very first transvection $t_{v\otimes f}$ we find. The reason is that all other transvections we are dealing with are obtained from the original one by either conjugating with elements of $G$ or taking commutators with another conjugate of the same transvection. If a transvection $t_{u\otimes g}$ is obtained from $t_{v\otimes f}$ in such a way, we can easily construct the generators of $T_{u\otimes g}$ of the form $\{t_{\theta^i u\otimes g},\ i = 0,\ldots,l-1\}$ with *the same* $\theta$ as for $T_{v\otimes f}$, "propagating" the original $\theta$. See Section 4-7.1 for details and pseudo–code..

In the next section we will construct transvections $t_{v_i\otimes g_j}$ $(i \neq j)$, where $\{v_1,\ldots,v_n\}$ and $\{g_1,\ldots,g_n\}$ are dual bases of $V$ and $V^*$ respectively. It turns out that knowing our $\theta$-generators of $T_{v\otimes f}$ is enough to construct the entire collection

$$\{t_{\theta^k v_i\otimes g_j} \mid k = 0,\ldots,l-1;\ i,j = 1,\ldots,n;\ i \neq j\}.$$

Hence for any transvection $t_{w\otimes h}$ we can compute the scalars $h(v_i)\cdot g_j(w)$, which is exactly what we need to recognize (write down the matrix corresponding to) *any* transvection $t_{w\otimes h}$.

## 4-7 Construction of a dual block pair

In this section we show how the properties of transvections can be used for constructing a set of transvections $\{t_{v_i\otimes g_j}\} \subset G$ such that their labels $v_i \otimes g_j$ form an $GF(q)$-basis in $V \otimes V^*$, and moreover, $\{v_i\}$ and $\{g_j\}$ are dual bases of $V$ and $V^*$ respectively. We keep assigning out labels to transvections in $G$ so that the labels are $\Theta$-consistent. In particular, when we claim that an identity involving transvections $t_1, t_2, \ldots$ holds in $G$, we mean that a similar identity holds for $t_1 = \Theta(\tau_1)$, $t_2 = \Theta(\tau_2), \ldots$ in $SL(n,q)$. More precisely, when a straight line program $\mathcal{R}$ applied to transvections $\tau_1, \tau_2, \ldots$ yields

$$\tau_{w\otimes h} = \mathcal{R}(\tau_1,\tau_2,\ldots) \in SL(n,q),$$

we assign the label $t_{w\otimes h}$ to $\mathcal{R}(t_1,t_2,\ldots) \in G$, on the strength of the fact that

$$\mathcal{R}(t_1,t_2,\ldots) = \mathcal{R}(\Theta(\tau_1),\Theta(\tau_2),\ldots) = \Theta(\mathcal{R}(\tau_1,\tau_2,\ldots)) = \Theta(\tau_{w\otimes h})\ .$$

**Theorem 4-7.1** *There exists an algorithm that, given a transvection $t_{v\otimes f} \in G$ and a ppd$(q,n-2)$ element $s \in G$ commuting with $t_{v\otimes f}$, constructs sets $B_\Gamma$, $B_\beta$ of transvections representing an $GF(q)$-basis $\{v_1,\ldots,v_n\}$ of $V$ and the corresponding dual basis $\{g_1,\ldots,g_n\}$ of $V^*$, namely*

$$B_\Gamma = \{t_{v_1\otimes g_n},\ldots,t_{v_{n-1}\otimes g_n}\}$$

*and*

$$B_\beta = \{t_{v_n \otimes g_1}, \ldots, t_{v_n \otimes g_{n-1}}\},$$

*in time* $O(q[\mu+\rho] + n^2 q\mu + n^3\varepsilon)$.

We call such a set of transvections a *dual block pair* in keeping with the notation of [CFL]. The sets $B_\Gamma$ and $B_\beta$ are the respective bases of the two maximal blocks of imprimitivity containing the original transvection $t = t_{v_1 \otimes g_n}$, if these blocks are considered as vector spaces over $GF(q)$.

Note that when we have these sets, we can construct any $t_{v_i \otimes g_j} \in G$ $(i \neq j)$ by simply taking the commutator

$$[t_{v_n \otimes g_j}, t_{v_i \otimes g_n}] = t_{v_i \otimes g_j}.$$

The proof by construction is carried out below.

**Theorem 4-7.2** *There exists an algorithm that, given the sets* $B_\Gamma$ *and* $B_\beta$ *of the previous theorem, and the full transvection group* $T_{v\otimes f} = T_{v_1 \otimes g_n}$*, can, for an arbitrary* $x \in G$*, produce a matrix representative of the coset* $\Theta^{-1}(x)$ *in* $SL(n,q)$ *under a fixed homomorphism* $\Theta : SL(n,q) \to G$*, in time* $O(n^2 q\mu + n^2\varepsilon)$.

In our constructive proof we use the generators of the full transvection group $T_{v\otimes f}$. Constructing them takes $O(q[\rho+\mu])$ time, as explained above.

### 4-7.1 Constructing a dual block pair for $V$, $V^*$

Given an original transvection $t = t_{v\otimes f} \in G$ and a *ppd*$(q, n-2)$ element $s \in G$ commuting with $t$, we can construct transvections representing a basis for the hyperplane $Ker(f)$, the axis if $t_{v\otimes f}$, as follows.

We know that $s \in G$ acts on transvections of $G$ by conjugation, and this action is related to the action of $s = \Theta(\sigma)$ on the subset of $V \otimes V^*$ that bijectively corresponds to transvections in $G$ by

$$t_{s(v\otimes f)} = t^s_{v\otimes f} = \Theta(\tau_{v\sigma\otimes\sigma^* f}) = \Theta(\tau^\sigma_{v\otimes f}).$$

Since it is more convenient to deal with an action on $V$ rather than that on $V \otimes V^*$, we are going to introduce some extra notation for use in this section.

First of all, notice that from $st = ts$ it follows that

$$t^s_{v\otimes f} = t_{v\otimes f} \quad \Leftrightarrow \quad \sigma(v \otimes f) = v \otimes f \ .$$

Notice that $\sigma \in SL(n,q)$ acts naturally on the one-dimensional spaces of $V$ (i.e. on the projective space $P(V)$), and its dual $\sigma^*$ acts on the one-dimensional spaces of $V^*$ (i.e. on $P(V^*)$). In particular, $\sigma$ and $\sigma^*$ fix $\langle v\rangle$ and $\langle f\rangle$ respectively.

The action of $\sigma$ on $\Gamma(f) = \{\tau_{u\otimes f} \mid u \in V,\ f(u) = 0,\ u \neq 0\}$ is therefore well-defined, since $\Gamma(f)^\sigma \subset \Gamma(f)$, and corresponds to its action on $\{u \otimes f \mid u \in Ker(f), u \neq 0\}$, which is also well-defined, since the latter is mapped to itself by $\sigma$. This allows us to define the action of $\sigma$ on $Ker(f) \subset V$ by letting

$$\sigma(u \otimes f) = (u\sigma) \otimes f \quad \forall u \in Ker(f).$$

The "functional part" $f$ of the tensor product being fixed, this provides a well-defined result $u\sigma$ of applying $\sigma$ to any $u \in Ker(f)$. This is an $(n-1)$-dimensional linear representation of $\langle\sigma\rangle \subset PSL(n,q)$, $\langle s\rangle \subset G$.

Now that $\sigma$ acts on the linear space $Ker(f)$, it is easy to see that it stabilizes the 1-dimensional subspace $\langle v\rangle \subset Ker(f)$. Observe that

$$Ker(f) = \langle v\rangle \oplus W, \ \text{ where } \ W = [Ker(f),\sigma] = \{u - u\sigma \mid u \in Ker(f)\}.$$

By Lemma 4-4.4, $\sigma$ acts irreducibly on $W$ (as $\dim W = n-2$), hence for any $u \in W$, $\langle u\rangle \neq \langle v\rangle$,

$$\{v, u, u\sigma, \ldots, u\sigma^{n-3}\}, \quad u \in W,\ \langle u\rangle \neq \langle v\rangle$$

is a basis for $Ker(f)$.

Thus in $SL(n,q)$, with $u \in W$, $u \notin \langle v \rangle$ as above, the set

$$\{\tau_{v\otimes f}, \tau_{u\otimes f}, \tau^{\sigma}_{u\otimes f}, \ldots, \tau^{\sigma^{n-3}}_{u\otimes f}\}$$

represents an $GF(q)$-basis of $Ker(f) \in V$ (and also forms a basis for $\Gamma(f) = \{\tau_{u\otimes f} \mid u \in V, f(u) = 0, u \neq 0\}$ (one of the two maximal blocks of imprimitivity that contain $t_{v\otimes f}$), if the latter is considered as a linear space over $GF(q)$.) By our $\Theta$-consistency, so does the set

$$B_\Gamma = \{t_{v\otimes f}, t_{u\otimes f}, t^{s}_{u\otimes f}, \ldots, t^{s^{n-3}}_{u\otimes f}\} \subset G.$$

The transvection $t_{u\otimes f}$ with $u$ as above can be found as follows. Apply `FindVect` to the original transvection $t_{v\otimes f}$ to find a transvection $t_{u'\otimes f}$ with the center $u' \notin \langle v \rangle$. Then

$$t_{u'\otimes f} \cdot (t^{s}_{u'\otimes f})^{-1} = t_{u'-u'\sigma\otimes f},$$

where $u' - u'\sigma \in W$ as desired, so the resulting transvection can be taken as $t_{u\otimes f}$ in the above construction.

Notice that for the purpose of getting bases for $Ker(f)$ and $\Gamma(f)$ we could use $t_{u'\otimes f}$ output by `FindVect` directly. Indeed, $u' - u \in \langle v \rangle$, and $\sigma$ fixes $\langle v \rangle$, so that $\{v, u', u'\sigma, \ldots, u'\sigma^{n-3}\}$ still yields a basis of $Ker(f)$.

Denote

$$\{t_{v_1\otimes f}, \ldots, t_{v_{n-1}\otimes f}\} := \{t_{v\otimes f}, t_{u\otimes f}, t^{s}_{u\otimes f}, \ldots, t^{s^{n-3}}_{u\otimes f}\} = B_\Gamma.$$

For what follows it will be necessary that we write out the straight line programs to these transvections from the original transvection $t = t_{v\otimes f} \in G$. Recalling how `FindVect` works, we get

$$t_{v_1\otimes f} = t, \tag{4.8}$$

$$t_{v_2\otimes f} = [t^x, t], \tag{4.9}$$

$$\tau_{v_3\otimes f} = [t^x, t]^s, \tag{4.10}$$

$$\ldots \tag{4.11}$$

$$t_{v_{n-1}\otimes f} = [t^x, t]^{s^{n-3}}, \tag{4.12}$$

where some $x \in G$ is an element found in the course of `FindVect` such that for $t_{v' \otimes f'} := t_{v \otimes f}^x$ we have $f(v') = 0$ and $f'(v) \neq 0$.

We want to solve the same problem for $\beta(v)$, the other maximal block of imprimitivity containing $t_{v \otimes f}$. Following the same reasoning as above, with appropriate changes from $V$ to $V^*$, we define the action of $\sigma^*$ on $(v)^\perp$, and see that this action is irreducible on its $(n-2)$ dimensional subspace, and fixes $\langle f \rangle$. Therefore we likewise construct a set of transvections $B'_\beta$ (which is not yet the $B_\beta$ we need for the dual block theorem, hence the prime $'$) that represents a basis for the hyperplane $(v)^\perp = \{h \in V^* \mid h(v) = 0\}$ as

$$B'_\beta = \{t_{v \otimes f}, t_{v \otimes h}, t_{v \otimes h}^s, \ldots, t_{v \otimes h}^{s^{n-3}}\},$$

where $t_{v \otimes h}$ is produced by `FindFunc`$(t_{v \otimes f})$. The straight line programs to the elements of $B'_\beta$ are the same as above, with the only difference being in the choice of the conjugating element $x \in G$, found in the process of `FindFunc`. Namely, $t_{v' \otimes f'} := t_{v \otimes f}^x$ must now satisfy $f(v') \neq 0$ and $f'(v) = 0$.

In what follows, we will need to perform the same construction not only for $\beta(v)$ and $(v)^\perp$ of the original transvection $t_{v \otimes f}$, but also for other transvections $t_{v' \otimes f'}$ obtained from the original one by conjugation. Fortunately, this is also possible, and the construction above goes through for $\beta(v')$ and $(v')^\perp$.

Indeed, take $t_{v' \otimes f'} := t_{v \otimes f}^y$. For any $y \in G$, the transvection $t_{v \otimes f}^y$ is stabilized by $ysy^{-1}$, which is also a $ppd(q, n-2)$ .

Following the same procedure but now starting from the transvection $t_{v' \otimes f'}$ and using the action of the $ppd(q, n-2)$ element $ysy^{-1}$ instead of $s$, we will get a set of transvections

$$\{t_{v' \otimes h_1}, t_{v' \otimes h_2}, \ldots, t_{v' \otimes h_{n-1}}\}.$$

representing the basis $\{h_1, \ldots, h_{n-1}\}$ of $(v')^\perp$. The straight line programs to $\{t_{v' \otimes h_i} : i = 1, \ldots, n-1\}$ are written as in (4.8) etc, with the change of $t$ to $t^y$ and $s$ to $ysy^{-1}$. The requirement on the conjugating element $x$ is that for $t_{v''\otimes f''} := (t^y)^x$ we must have $f'(v'') \neq 0$ and $f''(v') = 0$

We refer to the procedures just described as `Build-block-basis` and `Build-block-basis-1`.

We now want to construct the sets of transvection representing *dual* bases for $V$, $V^*$,

$$\{v_1, v_2, \ldots, v_{n-1}, v_n\} \text{ and } \{g_1, g_2, \ldots, g_{n-1}, g_n\},$$

so that $g_i(v_j) = \delta_{ij}$, $i, j = 1, \ldots, n$, and we set $v_1 = v$, $g_n = f$.

First, we locate a vector $v_n$ outside the span of $\{v_1, v_2, \ldots, v_{n-1}\}$ and then construct the set of transvections $\{\tau_{v_n\otimes g_1}, \ldots, \tau_{v_n\otimes g_{n-1}}\}$ with $\{g_i\}$ as above.

There is a good chance, $\sim 1 - \frac{1}{q}$, that, for a randomly chosen $x \in G$, $\tau_{v'\otimes f'} := \tau^x_{v\otimes f}$ is such that $f(v') \neq 0$ (see Lemma 4-5.1). This condition can be verified by procedure `IsZero` . Set $v_n = v'$ and construct (by procedure `Build-block-basis-1`) some basis $\{t_{v_n\otimes h_1}, \ldots, t_{v_n\otimes h_{n-1}}\}$ for $(v_n)^\perp$. Recalling how `FindFunc` works, we see that for suitable elements $y, x \in G$ chosen as explained above,

$$\begin{aligned} t_{v_n\otimes h_1} &= t^y, \\ t_{v_n\otimes h_i} &= [t^{yx}, t^y]^{s^{i-2}},\ i = 2, \ldots, n-1 \ . \end{aligned}$$

What follows is simply expressing one basis of $(v_n)^\perp$ in terms of another, $\{g_i\}$ via $\{h_i\}$. Since in our notation $\tau_{\alpha v\otimes f} \equiv \tau_{v\otimes\alpha f}$, and $t_{\alpha v\otimes f} \equiv t_{v\otimes\alpha f}$, we can set $f(v_n) = 1$ without loss of generality, and denote $g_n := f$ to keep the numbering of our basis vectors uniform.

We can find constants $f(v_n) \cdot h_i(v_j)$, i.e simply $h_i(v_j)$, by calling

$$\texttt{FindConst}(t_{v_j\otimes f}, t_{v_n\otimes h_i}, T_{v_j\otimes f}),\ i, j = 1, \ldots, n-1.$$

The full transvection groups $T_{v_j\otimes f}$ necessary for computing in $GF(q)$ as performed by `FindConst`, are easy to obtain. Indeed, recalling the straight line programs that lead to $\{t_{v_i\otimes f},\ i =$

$1, \ldots, n-1\}$ and starting from a transvection $t' = t_{\alpha v \otimes f}$ with any $\alpha \in GF(q)$ instead of the original $t_{v \otimes f}$, we get through the same straight line programs:

$$t_{\alpha v_1 \otimes f} := t', \tag{4.13}$$

$$t_{\alpha v_2 \otimes f} := [t^x, t'], \tag{4.14}$$

$$t_{\alpha v_3 \otimes f} = [t^x, t']^s, \tag{4.15}$$

$$\ldots \qquad \ldots$$

$$t_{\alpha v_{n-1} \otimes f} := [t^x, t']^{s^{n-3}}. \tag{4.16}$$

In particular, running these straight line programs for each of the generators $t_{\theta^k v \otimes f}$ of $T_{v \otimes f}$ we can obtain the full set

$$\{t_{\theta^k v_i \otimes f} : k = 0, \ldots, l-1,\ i = 1, \ldots, n-1\}.$$

Write $h_i = \sum_{j=1}^{n-1} a_{ij} g_j$, hence $h_i(v_j) = a_{ij}$ $(i, j = 1, \ldots, n-1)$ are all easily found.

Let $B = (b_{ij})$ be the inverse matrix to $A = (a_{ij})$, so

$$g_i = \sum_{j=1}^{n-1} b_{ij} h_j, \quad i = 1, \ldots, n-1$$

.

For the case of $q = p$ prime, $b_{ij} \in \mathbb{F}_p$, and we can immediately compute

$$t_{v_n \otimes g_i} = \prod_{j=1}^{n-1} t_{v_n \otimes h_j}^{b_{ij}}.$$

For $q$ non-prime, we have $b_{ij} \in GF(q)$, i.e. $b_{ij}$ are $l$-tuples with entries in $\mathbb{F}_p$. Write

$$b_{ij} = b_{ij,0} + b_{ij,1}\theta + \ldots + b_{ij,l-1}\theta^{l-1}, \quad b_{ij,d} \in \mathbb{F}_p,$$

and recall that

$$\{\theta^k h_i \mid k = 0, \ldots, l-1,\ i = 1, \ldots, n-1\}$$

is an $\mathbb{F}_p$-basis of $(v_n)^\perp$. Therefore

$$t_{v_n \otimes g_i} = \prod_{j=1}^{n-1} \prod_{d=0}^{l-1} (t_{v_n \otimes \theta^d h_j})^{b_{ij,d}}, \quad b_{ij,d} \in \mathbb{F}_p,$$

because of the identity in $SL(n,q)$

$$\prod_{j=1}^{n-1}\prod_{k=0}^{l-1}(\tau_{v_n\otimes\theta^k h_j})^{b_{ij,k}} = \prod_{j=1}^{n-1}\tau_{v_n\otimes(\sum_k b_{ij,d}\theta^d)h_j} = \tau_{v_n\otimes(\sum_j b_{ij}h_j)}.$$

The dual block pair is constructed.

Notice that each transvection $t_{v_i\otimes g_j}$ in the dual block pair $B_\Gamma\cup B_\beta$ we can construct also the generators

$$\{t_{v_i\otimes g_j}, t_{\theta v_i\otimes g_j}, t_{\theta^2 v_i\otimes g_j},\ldots,t_{\theta^{l-1}v_i\otimes g_j}\},$$

of its full transvection group, thus obtaining the system

$$S' = \{t_{\theta^k v_i\otimes g_j} \mid t_{v_i\otimes g_j}\in B_\Gamma\cup B_\beta,\ k=0,\ldots,l-1\}\ .$$

The pseudo-code for the procedures described above may look as follows. We skip handling of (infrequent) failures of Las Vegas `FindVect` and `FindFunc` for the sake of simplicity. Moreover, when complexity may be reduced by storing powers of elements of $G$, we indicate the better complexity but do not show the details in the pseudo-code.

`Build-block-basis` [for a hyperplane of $V$]

*Input:* $(n,t,s,G)$, where $n$ is the dimension of the $G$, $t=t_{v\otimes f}$ is a transvection, $s$ is a *ppd*$(q,n-2)$ element of $G$ commuting with $t$.

*Output:* $\{t_{v_1\otimes f},\ldots,t_{v_{n-1}\otimes f}\}$, representation of a basis in $Ker(f)\subset V$ by transvections.

*Complexity:* $O(q[\rho+\mu]+n\mu)$.

Set $t_{v_1\otimes f}\leftarrow t$

Set $t_{v_2\otimes f}\leftarrow$ `FindVect`$(t)$

For $i\leftarrow 3$ to $n-1$

  Set $t_{v_i\otimes f}\leftarrow t^s_{v_{i-1}\otimes f}$

Return$(\{t_{v_1\otimes f},\ldots,t_{v_{n-1}\otimes f}\})$

`Build-block-basis-1` [for a hyperplane of $V^*$]

*Input:* $(n, t, t', s, G)$, where $n$ is the dimension of the $G$, $t = t_{v\otimes f}$ is a transvection, and $t' \in \Gamma(f)$ is produced by `FindVect` , $s$ is a *ppd*$(q, n-2)$ element of $G$ commuting with $t$.

*Output:* $\{t_{v\otimes h_1}, \ldots, t_{v\otimes h_{n-1}}\}$, representation of a basis in $(v)^\perp \subset V^*$ by transvections.

*Complexity:* $O(q[\rho + \mu] + n\mu)$.

`Set` $t_{v\otimes h_1} \leftarrow t$

`Set` $t_{v\otimes h_2} \leftarrow$ `FindFunc`$(t, t')$

`For` $i \leftarrow 3$ `to` $n-1$

  `Set` $t_{v\otimes h_i} \leftarrow t^s_{v\otimes h_{i-1}}$

`Return`$(\{t_{v\otimes h_1}, \ldots, t_{v\otimes h_{n-1}}\})$

`Build-dual-pair` [Using Gaussian elimination]

*Input:* $(n, t, s, G)$, where $n$ is the dimension of the $G$, $t = t_{v\otimes f}$ is a transvection, $s$ is a *ppd*$(q, n-2)$ element of $G$ commuting with $t$.

*Output:* $(B_0, B_1)$, where $B_0 = \{t_{v_1\otimes f}, \ldots, t_{v_{n-1}\otimes f}\}$, $B_1 = \{t_{v_n\otimes g_1}, \ldots, t_{v_n\otimes g_{n-1}}\}$ such that $v_1, \ldots, v_n$ form a basis of $V$, and $g_1, \ldots, g_n$ is the corresponding dual basis in $V^*$ (and $v_1 = v$, $g_n = f$).

*Complexity:* $O(q[\rho + \mu] + n^2 q\mu + n^3)$

[Construct the model of $GF(q)$, the full transvection group of $t$.]

`Set` $T_{v\otimes f} \leftarrow$ `Build-transvection-basis`$(t, G)$

[Construct $\{t_{v_1\otimes f}, \ldots, t_{v_{n-1}\otimes f}\}$ ]

`Set` $B_0 \leftarrow$ `Build-block-basis`$(n, t, s, G)$

`Set` $t_a \leftarrow B_0[2]$ [ The choice of $\Gamma(f)$ is needed further for consistency ]

[Find $y \in G$ s.t. $t_{v' \otimes f'} = t^y_{v \otimes f}, f(v') \neq 0$ ]

`Repeat`

  `Set` $y \leftarrow$ `Random`$(G)$

`Until IsZero`$(t^y, t) =$ "Case 2"

`Set` $t_{v_n \otimes h} \leftarrow t^y$

[Having chosen $v_n$, generate a basis for $(v_n)^{\perp}$ ]

`Set` $\{t_{v_n \otimes h_1}, \ldots, t_{v_n \otimes h_{n-1}}\} \leftarrow$ `Build-block-basis-1`$(n, t_{v_n \otimes h}, t^y_a, s^y, G)$.

[Now to construct $g_i, i = 1, \ldots, n-1$]

[Compute $\{t_{\theta^d v_i \otimes f} \mid d = 0, \ldots, l-1,\ i = 1, \ldots n-1\}$, see (4.13)–(4.16)]

`For` $i \leftarrow 1$ `to` $n-1$

  `Set` $T_{v_j \otimes f} \leftarrow \{t_{\theta^d v_i \otimes f} \mid d = 0, \ldots, l-1\}$

`For` $i \leftarrow 1$ `to` $n-1$

  `For` $j \leftarrow 1$ `to` $n-1$

    Set $a_{ij} \leftarrow$ `FindConst`$(t_{v_j \otimes f}, t_{v_n \otimes h_i}, T_{v_j \otimes f})$ [takes $O(n^2 q\mu)$ ]

`Set` $(b_{ij}) \leftarrow$ `Invert-matrix`$((a_{ij}))$

`For` $i \leftarrow 1$ `to` $n-1$

  `Set` $t_{v_n \otimes g_i} \leftarrow \prod_{j=1}^{n-1} \prod_{d=0}^{l-1} (t_{v_n \otimes \theta^d h_j})^{b_{ij,d}}$

`Set` $B_1 \leftarrow \{t_{v_n \otimes g_1}, \ldots, t_{v_n \otimes g_{n-1}}\}$

`Return`$(B_0, B_1)$

### 4-7.2 Establishing $\Theta : SL(n,q) \to G_0$

In this section we conclude the construction of the mapping $\Theta : G \to G_0$, modulo verification that $\Theta$ is indeed a homomorphism, which we defer until Section 7.

We have now constructed two sets of transvections in $G$,

$$B_\Gamma = \{t_{v_1 \otimes g_n}, \ldots, t_{v_{n-1} \otimes g_n}\}$$

and

$$B_\beta = \{t_{v_n \otimes g_1}, \ldots, t_{v_n \otimes g_{n-1}}\}$$

such that $v_1 = v, \ldots, v_n$ form a basis of $V$, and $g_1, \ldots, g_{n-1}, g_n = f$ form the corresponding dual basis in $V^*$, so that $g_i(v_j) = \delta_{ij}$, $i, j = 1, \ldots, n$. Recall that we started with the single element $t = t_{v \otimes f} \in G$, of which it was known only that it was a transvection (without any specific knowledge of the center $v$ or the axis functional $f$). The sets $B_\Gamma, B_\beta$ were constructed by straight line programs involving a *ppd*$(q, n-2)$ element $s \in G$ and random elements $x, y, \ldots \in G$ that were found to satisfy certain commutation relations with $t$ and $s$. The initial elements $t, s \in G$ were also found by random trials; their respective properties of being a transvection and a *ppd*$(q, n-2)$ element were certified by reasoning involving only their orders and commutation relations. The construction of the bases $\{t_{\theta^k v_i \otimes g_j},\ k = 0, \ldots, l-1\}$ of the full transvection groups $T_{v_i \otimes g_j}$ for each $t_{v_i \otimes g_j} \in B_\Gamma \cup B_\beta$ followed the same principle.

Having assigned the label $v \otimes f$ to the original transvection $t$, we assigned other labels to elements of $G$ only if those were consistent with a homomorphism $\Theta : SL(n, q) \to G$. Now the images under $\Theta$ are fixed for a system of transvections

$$\mathcal{S}' = \{\tau_{\theta^k v_i \otimes g_j} \mid \Theta(\tau_{v_i \otimes g_j}) = t_{v_i \otimes g_j} \in B_\Gamma \cup B_\beta,\ k = 0, \ldots, l-1\}$$

that generates $SL(n, q)$. Denote $G_0 = \langle S' \rangle$ the subgroup of $G$ generated by the set

$$S' = \Theta(\mathcal{S}') = \{t_{\theta^k v_i \otimes g_j} \mid t_{v_i \otimes g_j} \in B_\Gamma \cup B_\beta,\ k = 0, \ldots, l-1\}\ .$$

The homomorphism $\Theta : SL(n, q) \to G_0$ is completely determined by the mapping $\mathcal{S}' \mapsto S'$, provided that the assignment of images as $t_{\theta^k v_i \otimes g_j} = \Theta(\tau_{\theta^k v_i \otimes g_j})$ indeed defines a homomorphism. Before further steps can be made, we must verify that. This is done by checking a sufficient subset of reduced Steinberg-Curtis-Tits relations, following [BGKLP]. The details of this are presented in Section refsec-verify-relations.

Provided that $\Theta : SL(n,q) \to G_0$ is well–defined, we can compute the image of any $\xi \in SL(n,q)$ by factoring it over $\mathcal{S}' \subset SL(n,q)$ and declaring $\Theta(\xi)$ to be word in the corresponding elements of $S' \subset G$. This construction can be simplified by expanding the system $S'$ to a more conventional Chevalley system

$$S = \{ t_{\theta^k v_i \otimes g_j} \mid i \neq j,\ i,j = 1, \ldots, n,\ k = 0, \ldots, l-1 \ \}$$

by adding $\Theta$-consistently labeled elements $t_{v_i \otimes g_j}$ and $t_{\theta^k v_i \otimes g_j}$ for remaining all pairs $i \neq j$, $i,j = 1, \ldots, n$, constructed as

$$t_{v_i \otimes g_j} = [\, t_{v_n \otimes g_j}, t_{v_i \otimes g_n} ] \ ,$$

and

$$t_{\theta^k \, v_i \otimes g_j} = [\, t_{\theta^k \, v_n \otimes g_j}, t_{v_i \otimes g_n} ] = [\, t_{v_n \otimes g_j}, t_{\theta^k \, v_i \otimes g_n} ] \ .$$

The corresponding system $\mathcal{S} \in SL(n,q)$ consists of all elementary matrices $\{ \tau_{\theta^k v_i \otimes g_j} \}$ with some power of a fixed element $\theta \in GF(q)$ in the only non-zero off-diagonal position. Such elementary matrices are known to generate $SL(n,q)$ provided that $\theta$ lies in no proper subfield of $GF(q)$.

We now want to compute with matrices over $GF(q)$ representing elements of $SL(n,q)$. We therefore declare the basis $\{v_i\}$ as the standard basis of $V$. Transvections $\tau_{v_i \otimes g_j}$ in this basis are represented by elementary matrices $E_{ij}$, with 1 in the only non-zero off-diagonal position $(i,j)$, and thus

$$\Theta : \ \tau_{v_i \otimes g_j} \ \longmapsto \ t_{v_i \otimes g_j} \ , \qquad i \neq j,\ i,j = 1, \ldots, n$$

maps the matrix $E_{ij}$ to $t_{v_i \otimes g_j}$.

Denote by $E_{ij}(\alpha)$ the elementary matrix with 1's on the diagonal, $\alpha \in GF(q)$ in the position $(i,j)$, and zeros elsewhere. Since the automorphism group of $GF(q)$ acts transitively on all elements of $GF(q)$ that lie in no proper subfield thereof, we can choose an arbitrary element $\theta$ of this kind and declare $t_{\theta v\otimes f} = t_{\theta v_1 \otimes g_n}$ to be the image of the transvection $\tau_{\theta v_1 \otimes g_n}$ which corresponds to the elementary matrix $E_{1n}(\theta)$ with $\theta$ in the only non-zero off-diagonal position $(1, n)$. By construction of $\Theta$, verified by the reduced Steinberg-Curtis-Tits relations, the assignment

$$\Theta : \ \tau_{\theta v_i \otimes g_j} \ \longmapsto \ t_{\theta\, v_i \otimes g_j} \ , \qquad i \neq j, \ i, j = 1, \ldots, n \ .$$

results in the mapping

$$\Theta : \ \tau_{\theta^k v_i \otimes g_j} \ \longmapsto \ t_{\theta^k v_i \otimes g_j}, \qquad i \neq j, \ i, j = 1, \ldots, n, \ k = 0, \ldots, l-1.$$

which maps the elements $\tau_{\theta^k v_i \otimes g_j}$ with matrices $E_{ij}(\theta^k)$ into $t_{\theta^k v_i \otimes g_j}$.

Since any matrix in $SL(n, q)$ can be represented as a product of $E_{ij}(\theta^k)$, we can now compute the image $\Theta(\xi)$ of any element $\xi \in SL(n, q)$ given by its matrix $M_\xi$. We now proceed to a detailed description of this algorithm.

**Lemma 4-7.3** *Let $\xi \in SL(n, q)$ be defined by its matrix $M_\xi$ in our chosen standard basis $\{v_i\}$. Then $M_\xi$ can be represented as a product of $O(n^2)$ elementary matrices in the same basis. The image $\Theta(\xi) \in G$ can be constructed in $O(n^3\varepsilon + n^2\mu \log q)$ time.*

Indeed, by $O(n^2)$ row– and column–reducing operations, each one a left or a right multiplication by an elementary matrix, we can arrive at the Bruhat decomposition of $M_\xi$ as

$$M_\xi = B_1 \, J \, B_2,$$

where $J$ is a permutation matrix with signs, $\det J = 1$ (corresponding to an element of the Weyl group of $SL(n, q)$) and $B_1, B_2$ are upper–triangular matrices. The computation is analogous to ordinary Gaussian elimination and takes $O(n^3\varepsilon)$ time, where $\varepsilon \sim \log q$ is the bit cost of computing with elements of $GF(q)$.

To go from this decomposition to the representation of $M_\xi$ as a product of elementary matrices (which $\Theta$ maps to previously constructed elements of $G$), it is enough to observe the

Steinberg relations in $SL(n,q)$. We will write them for $SL(2,q)$, from which they are easily generalized (see [C]). Let

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \in SL(2,q), \qquad \gamma \neq 0 \ .$$

Then

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} 1 & (\alpha-1)\gamma^{-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix} \begin{pmatrix} 1 & (\delta-1)\gamma^{-1} \\ 0 & 1 \end{pmatrix} . \tag{4.17}$$

Similarly, for $\beta \neq 0$,

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ (\delta-1)\beta^{-1} & 1 \end{pmatrix} \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ (\alpha-1)\beta^{-1} & 1 \end{pmatrix} . \tag{4.18}$$

Finally, for $\beta = \gamma = 0$,

$$\begin{pmatrix} \alpha & 0 \\ 0 & \alpha^{-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \alpha^{-1}-1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \alpha-1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -\alpha^{-1} \\ 0 & 1 \end{pmatrix} , \tag{4.19}$$

in particular,

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} .$$

Since any element of $S_n$ can be represented as a product of $n$ transpositions, writing $J$ as a product of elementary matrices requires as most $3n$ of those. Also, according to (4.19), we need at most $4n$ elementary transvections to write any diagonal matrix and $n(n-1)/2$ to complete it to an upper-triangular matrix. Thus the desired decomposition of $M_\xi$ involves $O(n^2)$ elementary matrices. Given the system of generators $S = \{t_{\theta^k v_i \otimes g_j}\} \subset G_0$, the image of an elementary matrix $E_{ij}(\alpha)$ can be constructed as

$$t_{\alpha v_i \otimes g_j} = t_{(a_{l-1}\theta^{l-1}+\ldots+a_1\theta+a_0)v_i\otimes g_j} = \prod_{k=0}^{l-1} t^{a_k}_{\theta^k v_i \otimes g_j} , \quad a_k \in \mathbb{F}_p \ ,$$

for $\alpha = a_{l-1}\theta^{l-1} + \ldots + a_1\theta + a_0$ in $O(\mu \log q)$ time. Consequently, constructing $\Theta(\xi)$ out of the images of $O(n^2)$ such matrices will take $O(n^2 \mu \log q)$ time.

As a first application of this algorithm we construct the image $Z \subset G$ of the center $\mathcal{Z}(SL(n,q)) = \{\lambda \cdot I \mid \lambda^n = 1\}$. By the above lemma and (4.19), we can construct the images of all scalar matrices in time $O(qn\mu \log q)$.

It remains to verify that $\Theta$, already verified to be onto $G_0 = \langle S' \rangle = \langle S \rangle$, is onto $G$, i.e. $G = G_0$. For this we need to locate the elements of $SL(n,q)$ that map onto the original generators $\langle g_1, \ldots, g_s \rangle = G$. The procedure through which we attempt to construct the matrices of that are mapped to $g_i$ is the general procedure we present for recovering a matrix representative in the pre-image of an arbitrary $x \in G_0$. Let us now describe this procedure, assuming $\Theta : SL(n,q) \to G_0 = \langle S \rangle$ to be a well-defined epimorphism.

### 4-7.3 Establishing $\Theta^{-1}$ by an action on the dual block pair

Any transvection $t' \in G$ can be written as $t' = t_{w \otimes h}$ for some $w \in V$, $h \in V^*$. Write $w$ and $h$ in the bases $\{v_i\}$, $\{g_j\}$:

$$w = a_1 v_1 + \ldots + a_n v_n, \quad h = b_1 g_1 + \ldots + b_n g_n,$$

keeping in mind that for any $\lambda \neq 0$ the $2n$-tuples

$$(a_1, \ldots, a_n, b_1, \ldots, b_n) \quad \text{and} \quad (\lambda a_1, \ldots, \lambda a_n, \lambda^{-1} b_1, \ldots, \lambda^{-1} b_n) \qquad (*)$$

define the same transvection

$$t_{w \otimes h} = t_{(a_1 v_1 + \ldots + a_n v_n) \otimes (b_1 g_1 + \ldots + b_n g_n)}.$$

Using procedures `IsZero` and `FindConst` on $t_{w \otimes h}$ and the elements of the dual block pair above allows us to find the coordinates $\{a_i\}$ and $\{b_j\}$ of $w$ and $h$ (or, more exactly, of $w \otimes h$) *explicitly*, from a system of equations. In other words, the dual block pair together with the above procedures are enough to identify any transvection.

This, in turn, helps to solve the problem of recognizing an arbitrary element $x \in G_0$. Conjugating the transvections from the dual block pair by $x$, we can identify the resulting transvections, and write down the matrix corresponding to $x$ in the basis $\{v_i\}$ modulo a single scalar.

Indeed, let $\xi \in SL(n,q)$, $x = \Theta(\xi)$, then

$$t^x_{v_k\otimes g_n} = \Theta(\tau^\xi_{v_k\otimes g_n}) = \Theta(\tau_{v_k\xi\otimes\xi^* g_n}) = \Theta(\tau_{(a_1^{(k)}v_1+\ldots+a_n^{(k)}v_n)\otimes(b_1^{(k)}g_1+\ldots+b_n^{(k)}g_n)}),$$

and

$$t^x_{v_n\otimes g_1} = \Theta(\tau^\xi_{v_n\otimes g_1}) = \Theta(\tau_{v_n\xi\otimes\xi^* g_1}) = \Theta(\tau_{(a_1^{(n)}v_1+\ldots+a_n^{(n)}v_n)\otimes(b_1^{(n)}g_1+\ldots+b_n^{(n)}g_n)})\ .$$

As a matter of fact, the coordinates $a_i^{(k)}$, $b_j^{(k)}$ completely define the action of $x \in G$ by conjugation on the set of transvections of $G$. Thus, by analyzing this action we can hope to find a matrix representative $\xi$, $x = \Theta(\xi)$.

Special care must be taken when writing equations for this conjugation action of $x$. Namely, recall that when $G$ has a non-trivial center $Z = \mathcal{Z}(G)$, this center is exactly the kernel of the conjugation action. Therefore, equations that come from these actions cannot help us recognize an element of $Z$ (as it acts trivially), or to distinguish between the elements in the same coset of $G/Z$ (since they act exactly the same way).

Moreover, denote $Z_0 = \Theta^{-1}(Z)$, $Z_0 \subset SL(n,q)$. Clearly, $\Theta^{-1}(x)$ is a coset of $Z_0$ in $SL(n,q)$. Recall that the kernel of the action of $SL(n,q)$ on its transvections by conjugation is its *entire* center, i.e. the subgroup of all of its scalar matrices. Thus, any equations written in $SL(n,q)$ will give us the matrix of $\xi \in \Theta^{-1}(x)$ only up to an element of $\mathcal{Z}(SL(n,q))$, i.e. up to an arbitrary scalar constant, not just up to an element of $Z_0 = Ker(\Theta)$ as we would like.

This problem can be remedied by computing the pre-image of the entire $Z$ as suggested above. Thus we can use the action of $x$ on $B_\Gamma \cup B_\beta$ to distinguish between the elements of any two *different* cosets of $G/Z$, and then pinpoint $x$ precisely by using the pre-computed center $Z$.

In the next section, we will describe the procedure for recognizing an arbitrary transvection $t_{w\otimes h}$ given the dual block pair and the basic procedures `IsZero` and `FindConst`. This procedure will yield coordinates $\{a_i, b_j\}$ of $w \otimes h$. It will be used repeatedly to find the coordinates $a_i^{(k)}, b_j^{(k)}$ of

$$v_1\xi \otimes \xi^* g_n, \quad \ldots, \quad v_{n-1}\xi \otimes \xi^* g_n, \quad v_n\xi \otimes \xi^* g_1$$

that arise as the labels of the $2n-2$ new transvections produced by the action of $x$ on $B_\Gamma \cup B_\beta$. More precisely, let $x \in G_0$, $\xi \in \Theta^{-1}(x)$ and denote

$$t_{w_i \otimes h_i} := t^x_{v_i \otimes g_n} = \Theta(\tau^\xi_{v_i \otimes g_n}) \;, \qquad i = 1, \ldots, n-1$$

$$t_{w_n \otimes h_n} := t^x_{v_n \otimes g_1} = \Theta(\tau^\xi_{v_n \otimes g_1}) \;.$$

Then, due to $\Theta$-consistency of our labeling,

$$t_{w_i \otimes h_i} := t_{x(v_i \otimes g_n)} = t_{v_i \xi \otimes \xi^* g_n}, \qquad i = 1, \ldots, n-1$$

and

$$t_{w_n \otimes h_n} := t_{x(v_n \otimes g_1)} = t_{v_n \xi \otimes \xi^* g_1},$$

The coefficients of the matrix $M_\xi$ relative to $\{v_1, \ldots, v_n\}$ are determined from the information about $\xi$ provided by the coefficients of $v_i\xi \otimes \xi^* g_n$, $v_n\xi \otimes \xi^* g_1$ relative to the basis $\{v_i \otimes g_j\}$. That is, given transvections $t_{w_i \otimes h_i}$ and the sets $B_\Gamma, B_\beta$ as above, we actually need to express $w_i$ and $h_i$ as linear combinations of $\{v_1, \ldots, v_n\}$ and $\{g_1, \ldots, g_n\}$ for $1 \le i \le n-1$.

**Identifying a transvection**

Let $t_{w \otimes h}$ be an arbitrary transvection,

$$w = a_1 v_1 + \ldots + a_n v_n, \quad h = b_1 g_1 + \ldots + b_n g_n.$$

Denote $c_{ij} = a_i b_j$. Let us show what equations in $a_i$, $b_j$ may be obtained.

Observe that for $i, j \le n-1$ we can get

$$c_{in} = a_i\, b_n = g_i(w)\, h(v_n) = \texttt{FindConst}(t_{v_n \otimes g_i}, t_{w \otimes h}, T_{v_n \otimes g_i}),$$

and

$$c_{nj} = a_n\, b_j = g_n(w)\, h(v_j) = \texttt{FindConst}(t_{v_j \otimes g_n}, t_{w \otimes h}, T_{v_j \otimes g_n}).$$

Further, using

$$[\, t_{v_n \otimes g_i}, t_{v_j \otimes g_n}] = t_{v_j \otimes g_i}, \quad (i \neq j),$$

we can get for $i \neq j$, $i, j \leq n-1$,

$$c_{ij} = a_i\, b_j = g_i(w)\, h(v_j) = \texttt{FindConst}(t_{v_j \otimes g_i}, t_{w \otimes h}, T_{v_j \otimes g_i}) \ .$$

Hence, using `FindConst`, we can get any equation of the type

$$a_i\, b_j = c_{ij}, \quad i \neq j$$

with each $c_{ij}$ computable in $O(q\mu)$ time. Of these, $2n-1$ are enough to find all $a_i$, $b_j$ up to a single scalar, except one special case in which we have to repeat the computation.

At least one of the coefficients $a_i$ must be non-zero, and the same is true for $b_j$. Without loss of generality suppose that $a_1 = g_1(w) \neq 0$ and $b_n = h(v_n) \neq 0$ (we cannot have two coefficients of the *same* index as the only non-zeros, since we must have $h(w) = 0$ for $t_{w \otimes h}$ to be a transvection.) Verifying if this is the case is straightforward - we can check whether $b_n = h(v_n) = 0$ or $a_1 = g_1(w) = 0$ by calling $\texttt{IsZero}(t_{w \otimes h}, t_{v_n \otimes g_1})$, which will answer both questions. Then we obtain $2n-2$ equations

$$\begin{cases} a_i\, b_n = c_{in}, & i = 2, \ldots, n \\ a_1\, b_j = c_{1j}, & j = 1, \ldots, n-1 \end{cases} \tag{4.20}$$

i.e. for every $j = 2, \ldots, n$ either $a_1\, b_j = c_{1j} \neq 0$ or $b_j = 0$, and, analogously, for every $i = 1, \ldots, n-1$ either $a_i\, b_n = c_{in} \neq 0$ or $a_i = 0$.

Recall that for any $\lambda \neq 0$ the $2n$-tuples

$$(a_1, \ldots, a_n, b_1, \ldots, b_n) \text{ and } (\lambda a_1, \ldots, \lambda a_n, \lambda^{-1} b_1, \ldots, \lambda^{-1} b_n)$$

define the same transvection. Therefore, knowing that $b_n \neq 0$ we can choose the scalar $\lambda$ so that $b_n = 1$. Then we have

$$a_1 = c_{1n}, \quad a_2 = c_{2n}, \quad \ldots, \quad a_{n-1} = c_{n-1,n},$$

$$b_2 = \frac{c_{12}}{a_1}, \quad \ldots, \quad b_{n-1} = \frac{c_{1,n-1}}{a_1}, \quad b_{n-1} = 1,$$

where $a_1$ is known from the first equation.

It remains to find $a_n$ and $b_1$. We know that

$$0 = h(w) = \sum_{i=1}^{n} a_i b_i = c_{1n} b_1 + \sum_{i=2}^{n-1} a_i b_i + a_n,$$

hence

$$b_1 a_1 + a_n = -\sum_{i=2}^{n-1} \frac{c_{1i} c_{in}}{a_1} \qquad (**) \ .$$

Suppose we have found that for some $k$, $2 \leq k \leq n-1$, $a_k \neq 0$. Then we can obtain the equation $a_k b_1 = c_{k1}$ and find $b_1$, and then find $a_n$ from the above equation. Analogously, any non-zero $b_l$, $2 \leq l \leq n-1$ will yield $a_n$ via $a_1 b_l = c_{nl}$.

We encounter a difficulty in the case where $a_j = b_j = 0$, $j = 2, \ldots, n-1$. We can still get the equation $b_1 a_n = c_{n1}$. If $c_{n1} = 0$, we can check by calling `IsZero`$(t_{w\otimes h}, t_{v_1 \otimes g_n})$ whether $b_1 = 0$ or $a_n = 0$ , and get the other one from (**). Otherwise $b_1 a_n = c_{n1} \neq 0$, and we have a quadratic equation:

$$\frac{c_{1n} c_{n1}}{a_n} + a_n = R,$$

where $R$ is the right-hand side of (**). There will be two solutions for $a_n$ (and hence for $b_1$).

We need not solve the above quadratic equation to deal with this indeterminacy. Notice that that to complete the solution we only need to know the ratio $\gamma = a_n / a_1$. Use procedure `FindFunc` to obtain a transvection $t_{w \otimes h'}$, which has the same set of coordinates $\{a_i\}$ but different $\{b'_j\}$. With probability $\gg (1 - 1/q)$ some three of these $b'_i$ will be non-zero, in which case we can solve for $a_i$ and obtain the missing ratio $\gamma$.

Hence, in all cases except the last one, finding $\{a_i\}$, $\{b_j\}$ involves solving a system of $2n-1$ equations, which requires $O(n)$ field operations, and the latter will, with high probability, at most double the required time.

**Recovering the matrix representative of $\Theta^{-1}(x)$**

From equations described in the previous section we get the coordinates

$$a_i^{(k)},\ b_j^{(k)}, \qquad i,j,k = 1,\ldots,n\ .$$

of the elements $(B_\Gamma \cup B_\beta)^x$. For $\xi \in SL(n,q)$, $x = \Theta(\xi)$ we have

$$v_k\xi \otimes \xi^* g_n = (a_1^{(k)}v_1 + \ldots + a_n^{(k)}v_n) \otimes (b_1^{(k)}g_1 + \ldots + b_n^{(k)}g_n) =$$
$$= \mathbf{a}^{(k)} \otimes \mathbf{b}^{(k)}, \quad k = 1,\ldots,n-1$$

and

$$v_n\xi \otimes \xi^* g_1 = (a_1^{(n)}v_1 + \ldots + a_n^{(n)}v_n) \otimes (b_1^{(n)}g_1 + \ldots + b_n^{(n)}g_n) =$$
$$= \mathbf{a}^{(n)} \otimes \mathbf{b}^{(n)}.$$

Let us refer to the coordinate vector $\mathbf{a}^{(k)}$ (in the standard basis $\{v_i\}$) as the *vector part* of $\mathbf{a}^{(k)} \otimes \mathbf{b}^{(k)}$, and to $\mathbf{b}^{(k)}$ as its *functional part.* Both parts are defined up to a common constant (see (*) above). We also write $v_k\xi = \mathbf{a}^{(k)}$ etc. in the sense that in our chosen standard basis vector $v_k\xi$ has coordinates $\mathbf{a}^{(k)}$.

Notice that, in the first $n-1$ equations, the functional part is the same and equals to $\xi^* g_n$, so we have $\mathbf{b}^{(1)} = \ldots = \mathbf{b}^{(n-1)}$. Then the vector parts $\mathbf{a}^{(k)} = v_k\xi$ for $k = 1,\ldots,n-1$ will form the first $n-1$ columns of the matrix of some $\xi' \in SL(n,q)$ which differs from $\xi$ only by a central element.

Solving the equations for the coordinates of $v_n\xi \otimes \xi^* g_1$, we actually get a series of solutions $(\mu\mathbf{a}^{(n)}, \mu^{-1}\mathbf{b}^{(n)})$, parametrized by a scalar $\mu \in GF(q)$. Let us choose one of them, some

$(\mathbf{a}^{(n)}, \mathbf{b}^{(n)})$. There exists a $\lambda \in GF(q)$ such that $v_k\xi = \mathbf{a}^{(k)}$, $k = 1, \ldots, n-1$, and $v_n\xi = \lambda\mathbf{a}^{(n)}$. But the action of $\xi \in SL(n,q)$ on the dual bases $\{v_i\}$, $\{g_j\}$ of $V, V^*$ must satisfy

$$(\xi^* g_1)(v_1\xi) = g_1(v_1) = 1,$$

which gives us an equation for $\lambda$. In our coordinates, $\xi^* g_1 = \lambda^{-1}\mathbf{b}^{(n)}$, and above we have taken $v_1\xi = \mathbf{a}^{(1)}$, so

$$< \mathbf{b}^{(n)}, \mathbf{a}^{(1)} >= b_1^{(n)}a_1^{(1)} + \ldots + b_n^{(n)}a_n^{(1)} = \lambda.$$

Now we have a matrix $M = (a_i^{(k)})$ that describes the action of $x$ on $V \otimes V^*$.

The matrix $M$ we obtained from our system of equations corresponds in the chosen basis to an element $\xi' \in SL(n,q)$. We know that the matrix of $\xi$, $x = \Theta(\xi)$, is different from $M$ only by a scalar multiple $\kappa$, i.e. $\xi' = (\kappa \cdot I)\,\xi$, $\kappa \in GF(q)$, where $I$ is the identity in $SL(n,q)$. Thus we use the matrix $M$ to construct $x' = \Theta(\xi')$, and then construct $x' \cdot x^{-1} \in Z$. Clearly,

$$x' \cdot x^{-1} = \Theta(\xi' \cdot \xi^{-1}) = \Theta(\kappa \cdot I) \;\in\; Z = \mathcal{Z}(G) = \Theta(\mathcal{Z}(SL(n,q))) \;.$$

Earlier we have constructed the set $Z$ elements-wise, so at the cost of at most $|Z| < q$ comparisons we can identify $\kappa \in GF(q)$ and finally obtain the desired matrix $M_\xi = \kappa^{-1} \cdot M$ of $\xi \in SL(n,q)$ such that $x = \Theta(\xi)$.

We now estimate the complexity of these algorithms. The header for the procedure `Find-image` that computes a matrix representative for the coset of $\xi$ is

`Find-image` [Find a matrix representative using the dual block pair.]

> *Input:* $(x, B_0, B_1)$, where $x \in G$, $(B_0, B_1)$ is the pre-computed dual block pair.
>
> *Output:* Matrix $(\, a_i^{(k)} \,)$ of $\Theta(x)$ modulo a constant in $GF(q)$.
>
> *Complexity:* With constant probability, $O(n^2 q\mu + n^2\varepsilon)$.

The matrix entries $a_i^{(k)}$, $i = 1, \ldots, n$ are found from $2n-1$ equations for each $k$, therefore the time to obtain all required equations is, with constant probability, $O(n^2 q\mu)$, the time to solve all of them $O(n^2\varepsilon)$, where $\varepsilon$ is the time required for a field operation. Considering the number of bits required to store elements of $\mathbb{F}_p$, we have that, at least, $\varepsilon \sim \log q$.

The full recognition algorithm consists of the following steps:

- find the initial transvection.
- pre-compute the dual block pair $B_\Gamma$, $B_\beta$, in $O(q[\rho + \mu] + n^2 q\mu + n^3\varepsilon)$ time.
- pre-compute the bases of the full transvection groups $T_{v_i \otimes g_j}$ of the dual block pair transvections, in $O(q[\rho + \mu] + n\mu)$ time. An $\mathbb{F}_p$-basis is found for one full transvection group and then carried to other groups by conjugation and commutation in $G$. This completely defines the homomorphism $\Theta$.
- pre-compute the center $Z = \mathcal{Z}(G)$, in $O(\mu[q \log n + n])$ time
- apply `FindImage`$(x)$, in $O(n^2 q\mu)$ time per element of $x \in G$ to get the matrix $M$ of $\xi \in SL(n, q)$, $\xi = \Theta(x)$, up to a central element of $SL(n, q)$, i.e. a scalar matrix multiplier.
- find the image $\Theta(\xi')$ of the element $\xi' \in SL(n, q)$ as defined by $M$, and construct the central element

  $$\Theta(\xi'\xi^{-1}) := \Theta(\xi') \cdot \Theta(\xi)^{-1} \in Z,$$

  in $O(n^3\varepsilon + nq\mu \log q)$ time, using the generating system $S' \subset G$ composed of elementary transvections from $B_\Gamma \cup B_\beta$ and $T_{v_i \otimes g_j}$.
- identify $\xi'\xi^{-1} \in \mathcal{Z}(SL(n, q))$ by its image, via a full search over $Z$.

To make our algorithm Las Vegas, we need two extra verification steps. We first need to verify that the mapping $\Theta$ defined by mapping a generating system of elementary matrices of

$SL(n,q)$ to $S'$ is indeed a homomorphism into its image $G_0 = \langle S' \rangle$. This step is performed right after the dual pair is constructed. The second step checks that $\Theta$ is onto $G$ itself, i.e. that $G_0 = G$, by expressing the original generators of $G$ via $S'$. The procedure we need for the second step is the general recognition procedure described above, with the additional verification of the result at the end. The first check involves a version of reduced Steinberg-Curtis-Tits relations, and is described in the next section.

## 4-8 Verifying the results and making the recognition algorithm Las Vegas.

Let $\{g_1, \ldots, g_s\}$ be the original generators of $G$. After a dual block pair $(B_\Gamma, B_\beta)$ for $G$ is obtained, and a basis for the full transvection group

$$T_{v_1 \otimes g_n} = \{t_{\alpha v \otimes f} \mid \alpha \in GF(q)\}$$

is constructed, and propagated by conjugation to the full transvection groups $T_{v_i \otimes g_j}$, $t_{v_i \otimes g_j} \in B_\Gamma \cup B_\beta$, we have a set

$$S' = \{t_{\theta^k v_i \otimes g_j} \mid t_{v_i \otimes g_j} \in B_\Gamma \cup B_\beta,\ k = 0, \ldots, l-1\},$$

easily extended to

$$S = \{t_{\theta^k v_i \otimes g_j} \mid k = 0, \ldots, l-1,\ 1 \leq i \neq j \leq n\}. \tag{4.21}$$

We must verify that these new generators satisfy the appropriate Chevalley group relations to generate a homomorphic image of $SL(n,q)$.

In order to keep the cost of this verification below that of our recognition algorithm we will make use of a short presentation of $SL(n,q)$ by generators and relations.

The set $S$ is a subset of the standard set of generators for $SL(n,q)$ as a Chevalley group, introduced by Steinberg. Therefore we could check the full Steinberg relations between these generators. Instead, we will actually check a reduced set of relations to save time. This will prove that our black box group $G$ indeed contains a subgroup $G_0$ homomorphic to $SL(n,q)$. We reduce the set $S$ to a smaller set $S''$, and it is for $S''$ that we check appropriate relations.

Let us now describe the reduced set $S''$ and the relations we have to check for the elements of this set to make sure that the group $G_0 = \langle S'' \rangle$ is a homomorphic image of $SL(n,q)$.

The Steinberg relations for $S$ are largely excessive, so we will use the reduced version of Curtis-Steinberg-Tits relations, as described in [BGKLP]. Below we follow the notation of [BGKLP].

Let us denote

$$x_{ij}(\alpha) = t_{\alpha v_i \otimes g_j} = \Theta(\tau_{\alpha v_i \otimes g_j})$$

to match the standard notation for Chevalley groups. Our full transvection groups are, of course, the root groups in this terminology.

The Steinberg relations for the set $\{x_{ij}(\alpha) \mid 1 \leq i,j \leq n,\ i \neq j,\ \alpha \in GF(q)\}$ of $n(n-1)q$ generators of $SL(n,q)$, $n \geq 3$ are

$$\begin{aligned}
x_{ij}(\alpha+\beta) &= x_{ij}(\alpha)\, x_{ij}(\beta) \\
[x_{ij}(\alpha),\, x_{jk}(\beta)] &= x_{ik}(\alpha\beta) \qquad \text{if } \ i,j,k \ \text{ are different} \\
[x_{ij}(\alpha),\, x_{km}(\beta)] &= 1 \qquad \text{if } \ j \neq k,\, i \neq m.
\end{aligned}$$

A significantly smaller subset of these relations is sufficient for a presentation of $SL(n,q)$. It is the Curtis-Steinberg-Tits presentation, which, for $SL(n,q)$, uses only the generators $x_{ij}(\alpha)$ with $|i-j| \leq 2$, and only the following relations of each of the above three types:

1. relations of the first type for $|i-j| \leq 2$,
2. relations of the second type whenever $i,j,k$ are three consequtive integers (in some order),
3. relations of the third type whenever $i,j,k,m$ all differ by at most 1 from some fixed integer value.

This produces a presentation with $O(nq)$ generators and $O(n^2q^2)$ relations.

Still, what we have available is a smaller set of generators, namely not all of $x_{ij}(\alpha)$, but only $x_{ij}(\theta^k)$, $k = 0, \ldots, l-1$, where the first $l$ powers of $\theta \in GF(q)^\times$ form an $\mathbb{F}_p$-basis for $GF(q)$. For these we can use the reduced presentation from [BGKLP], and that will be our reduced set of generators $S''$.

Recall that for any $\alpha \in GF(q)$ we can write

$$\alpha = b_{l-1}\theta^{l-1} + \ldots + b_1\theta + b_0, \qquad b_i \in \mathbb{F}_p$$

and, in the Chevalley group notation above,

$$x_{ij}(\alpha) = \prod_{k=0}^{l-1} x_{ij}(\theta^k)^{b_k}. \tag{4.22}$$

The relations that the set

$$S'' = \{x_{ij}(\theta^k) \mid |i-j| \leq 2,\, 1 \leq i \neq j \leq n,\, 0 \leq k \leq l-1\}$$

should satisfy to generate $SL(n,q)$ are

$$\begin{aligned}
x_{ij}(\theta^\nu)^p &= 1, \\
[x_{ij}(\theta^\nu), x_{ij}(\theta^\mu)] &= 1, \\
[x_{ij}(\theta^\nu),\, x_{jk}(\theta^\mu)] &= x_{ik}(\theta^{\nu+\mu}) \qquad \text{if } \ i,j,k \ \text{ are consecutive integers} \\
[x_{ij}(\theta^\nu),\, x_{km}(\theta^\mu)] &= 1 \qquad \text{if } \ i \neq j,\, k \neq m,
\end{aligned}$$

where $i, j, k, m$ in the last relation all differ by at mosts one from some fixed integer value, and where $x_{ik}(\theta^{\nu+\mu})$ for $\nu + \mu > l - 1$ must be understood as the right-hand side of equation (4.22).

Checking these relations will take the total of $O(l^2n^2 \log q) = O(l^3n^2 \log p)$ multiplications (by introducing additional generators and relations this cost can be further reduced to $O(n^2 l \log p)$). In other words, we can verify that our black box group contains a subgroup $G_0$ which is a homomorphic image of $SL(n,q)$ in time comparable with that required for the recognition algorithm itself. It only remains to check if the center of $G_0$ is non-empty.

Recall that we used to construct central elements of $G$ as images of elementary matrices in $SL(n,q)$. Now let us do the same using the elements of $S''$. Denote

$$\begin{aligned} w_{ij}(\alpha) &= x_{ij}(\alpha)x_{ji}(-\alpha^{-1})x_{ij}(\alpha)\,, \\ h_{ij}(\alpha) &= w_{ij}(\alpha)w_{ij}(1)^{-1}. \end{aligned}$$

The elements $h_{ij}(\alpha)$ ($|i-j| \leq 2$) are all diagonal, and they generate the Cartan subgroup of $G$ which is abelian and contains $\mathcal{Z}(G)$. General Chevalley theory shows that for $SL(n,q)$ the center consists of exactly those elements

$$h_{12}(\alpha_1)\cdots h_{k,k+1}(\alpha_k)\cdots h_{n-1,n}(\alpha_{n-1})$$

for which $\alpha_i = \alpha_1^i,\ 2 \leq i \leq n-1$, and $\alpha_1^n = 1$. Constructing and checking such elements for being the identity will take $O(|\mathcal{Z}(G)|\,\mu) = O(n\mu)$ time.

# Bibliography


[Asch] M. Aschbacher, “On the maximal subgroups of the finite classical groups”, *Invent. Math.* **76**, (1984), pp. 496–514

[Ba1] L. Babai, ”Local expansion of vertex-transitive graphs and random generation in finte groups”, *Proc.* $23^{rd}$ *ACM STOC*, pp. 164–174

[Ba2] L. Babai, ”Randomization in group algorithms: Conceptual questions”, in: *Groups and Computation II*, L. Finkelstein, W.M. Kantor, eds., *DIMACS Series in Discrete Mathematics and Computer Science* **28**, AMS (1997)

[BaSz] L. Babail, E. Szémeredi, ”On the complexity of matrix group problems I”, *Proc.* $25^{th}$ *IEEE Symp. Found. Comp. Science* (1984), IEEE Press, pp. 229–240

[Be] R. Beals. “Toward polynomial time algorithms for matrix groups”, in: *Groups and Computation II*, L. Finkelstein, W.M. Kantor, eds., *DIMACS Series in Discrete Mathematics and Computer Science* **28**, AMS (1997)

[BB] L. Babai and R. Beals, (1993), “Las Vegas Algorithms for Matrix Groups”, *Proc.* $24^{th}$ *IEEE FOCS*, pp. 427–436.

[BGKLP] L. Babai, A.J. Goodman, W.M. Kantor, E.M. Luks, P.P. Palfy, “Short Presentations for Finite Groups”, (1995)

[BP] S. Bratus, I. Pak, "Fast constructive recognition of a black box group isomorphic to $S_n$ or $A_n$ using Goldbach's Conjecture", submitted for publication (1997)

[C] Roger. W. Carter, "Simple Groups of Lie Type", *Wiley*, (1972)

[CC] P.J. Cameron and J. Cannon, "Fast recognition of doubly transitive groups. Computational groups theory, Part 2", J. Symbolic Comput, 12 no. 4-5 (1991), pp. 459–474

[CF] G. Cooperman, L. Finkelstein, "Topics in Computing with Large Groups of Matrices", *to appear*

[CFL] G. Cooperman, L. Finkelstein, and S. Linton, "Constructive recognition of a black box group isomorphic to $GL(n, 2)$", in: *Groups and Computation II*, L. Finkelstein, W.M. Kantor, eds., *DIMACS Series in Discrete Mathematics and Computer Science* **28**, AMS (1997), pp. 85–100

[CFTY] G. Cooperman, L. Finkelstein, M. Tselman and B. York, "Constructing permutation representations for matrix groups", *J. of Symbolic Computation* **24**, (1997), pp. 1–18

[CFDi] G. Cooperman and L. Finkelstein, "Combinatorial Tools for Computational Group Theory", *Proceedings of DIMACS Workshop on Groups and Computation*, DIMACS-AMS **11**, 1993, pp. 53–86, L. Finkelstein and W.M. Kantor (eds.), AMS, Providence, RI.

[CL1] F. Celler and C.R. Leedham-Green, "A non-constructive recognition algorithm for the special linear and other classical groups", in *Groups and Computation II*, L. Finkelstein, W.M. Kantor, eds., *DIMACS Series in Discrete Mathematics and Computer Science* **28**, AMS (1997), pp. 273–296

[CL2] F. Celler and C.R. Leedham-Green, "Calculating the Order of an Invertible Matrix", in: *Groups and Computation II*, L. Finkelstein, W.M. Kantor, eds., *DIMACS Series in Discrete Mathematics and Computer Science* **28**, AMS (1997), pp. 55–60

[CLMNO] F. Celler, C.R. Leedham-Green, S.H. Murray, A.C. Niemeyer, and E.A. O'Brien, "Generating random elements of a finite group", Comm. Algebra **23** (1995), pp. 4931–4948

[CM] H.S.M. Coxeter, W.O.J. Moser, "Generators and relations for discrete groups" (Fourth Edition), *Springer Verlag*, New York, 1980

[FHL] Merrick Furst, John Hopcroft, Eugene Luks, "Polynomial–Time algorithms for Permutation Groups", *Proc 21st IEEE Symp. Found. Comp. Sci.*, (1980), pp. 36–41

[H] B. Huppert, "Endliche Gruppen I", *Springer*, Berlin (1967)

[HL] G.H. Hardy, J.E. Littlewood, "Some problems of 'Partitio Numerorum'; III: On the expression of a number as a sum of primes", *Acta Math. vol. 44* (1922), pp. 1–70

[KS] William Kantor and Ákos Seress, "Black Box Classical Groups", *to appear.*

[L] E. M. Luks, "Computing the Composition Factors of a Permutation Groups in Polynomial Time", *Combinatorica*, **7**(1), (1987), pp 87–99

[MV] H.C. Montgomery, R.C. Vaughan, "The exceptional set in Goldbach problem", *Acta Arith. vol. 27* (1975), pp. 353–370

[NeP] P.M. Neumann and C.E. Praeger, "A recognition algorithms for special linear groups", *Proc. London Math. Soc.* **65** (1992), pp. 555–603

[NieP] A.C. Niemeyer and C.E. Praeger, "A recognition algorithms for classical groups", to appear in *Proc. London Math. Soc.*

[NiP] A.C. Niemeyer and C.E. Praeger, "Implementing a recognition algorithm for classical group", *Groups and Computation II*, L. Finkelstein and W.M. Kantor, eds., *DIMACS Series in Discrete Mathematics and Computer Science* **28**, AMS (1997), pp. 273–296

[Ra] Ramaré, "On Schnirelman's Constant", *Ann. Scuola. Norm. Sup. Pisa Cl. Sci.* **22:4** (1995), pp. 645-706

[Ri] P. Ribenboim, *"The new book of prime numbers"*, *Springer Verlag*, New York, 1995

[Si] C.C. Sims, "Computation with Permutation Groups", in *Proc. Second Symposium on Symbolic and Algebraic Manipulation*, edited by S.R. Petrick, ACM Press, New York, 1971, pp. 23–28

[Sy] J.J. Sylvester, "On the partition of an even number into two primes", *Proc. London Math. Soc.* Series 1, **4** (1871), pp. 4–6

[V] I.M. Vinogradow, "Representation of an odd number as a sum of three primes", *Dokl. Akad. Nauk SSSR* **15** (1937), pp. 291–294

[W] Wang Yuan, "Goldbach Conjecture", *World Scientific Publishing Co.*, (1984)

[We] M. Weller, "Construction of Large Permutation Representations for Matrix Groups", preprint 17 (1997), Institut für Experimentelle Mathematik, Essen, Germany