NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF MATHEMATICS

MADALENA BRAVO FERRO DA ASCENSÃO

BSc in Mathematics Applied to Economics and Management

CRYPTOGRAPHIC KEY EXCHANGE PROTOCOLS: ALGEBRAIC STRUCTURES, STICKEL'S PROTOCOL, AND MONOID INVESTIGATIONS

MASTER IN MATHEMATICS AND APPLICATIONS
SPECIALIZATION IN PURE MATHEMATICS

NOVA University Lisbon
22 May, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
MATHEMATICS

CRYPTOGRAPHIC KEY EXCHANGE PROTOCOLS: ALGEBRAIC STRUCTURES, STICKEL'S PROTOCOL, AND MONOID INVESTIGATIONS

MADALENA BRAVO FERRO DA ASCENSÃO

BSc in Mathematics Applied to Economics and Management

Adviser: Doutor António Malheiro
Full Professor, NOVA University Lisbon

Co-adviser: Doutor André Carvalho
Junior Researcher, University of Porto

Examination Committee

President: Doutora Magda Stela de Jesus Rebelo
Associate Professor, NOVA University Lisbon

Arguer: Doutor João Jorge Ribeiro Soares Gonçalves de Araújo
Full Professor, NOVA University Lisbon

MASTER IN MATHEMATICS AND APPLICATIONS
SPECIALIZATION IN PURE MATHEMATICS

NOVA University Lisbon
22 May, 2024

Cryptographic Key Exchange Protocols: Algebraic Structures, Stickel's Protocol, and Monoid Investigations

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.11) [22].

Abstract

This thesis provides a comprehensive examination of cryptographic key-exchange protocols leveraging algebraic structures for key encoding. The focal point of our investigation is Stickel's protocol. We not only elucidate the protocol but also implement it in Python. Subsequently, we conduct thorough testing and benchmark its execution time, revealing a possible exponential relationship with the input size. This observation aligns with the findings of Myasnikov, Shpilrain, and Ushakov.

The research extends to the exploration of East's work, which offers various presentations of the partition monoid. Furthermore, we delve into specific submonoids, namely $\mathcal{L}_n$, $\mathcal{R}_n$, and the inverse symmetric monoid $\mathcal{I}_n$. For $\mathcal{I}_n$ we conclude that the presentation is not confluent. We ended the study by inferring that the presentation of the partition monoid by East is not complete.

The thesis culminates in the formulation of a complete presentation for the planar rook monoid, a notable submonoid of the partition monoid. This investigation is motivated by the consideration of partition monoid as potential alternative platform for Stickel's protocol. The synthesis of these cryptographic protocols and algebraic structures contributes to a deeper understanding of the interplay between algebraic structures and cryptographic key exchange, paving the way for potential advancements in secure communication protocols.

Keywords: Cryptography, Partition monoid, Planar rook monoid, Rewriting systems, Stickel protocol

Resumo

Esta tese fornece uma análise abrangente de protocolos de troca de chaves criptográficas que utilizam estruturas algébricas para codificação de chaves. O ponto central da nossa investigação é o protocolo de Stickel. Não só elucidamos o protocolo, mas também o implementamos em Python. Posteriormente, realizámos testes abrangentes e avaliámos o tempo de execução, revelando uma possível relação exponencial com o tamanho do input. Esta observação está alinhada com as descobertas de Myasnikov, Shpilrain e Ushakov.

A investigação estende-se à exploração do trabalho de East, que oferece várias apresentações do monoide de partição. Além disso, aprofundámos submonoides específicos, nomeadamente $\mathcal{L}_n$, $\mathcal{R}_n$, e o monoide simétrico inverso $\mathcal{I}_n$. Para $\mathcal{I}_n$, concluímos que a apresentação não é confluente. Terminámos o estudo, inferindo que a apresentação do monoide de partição por East não é completa.

A tese culmina na formulação de uma apresentação completa para o monoide de torre planar, um notável submonoide do monoide de partição. Esta investigação é motivada pela consideração do monoide de partição como potencial plataforma alternativa para o protocolo de Stickel. A síntese desses protocolos criptográficos e estruturas algébricas contribui para uma compreensão mais profunda da interação entre estruturas algébricas e troca de chaves criptográficas, abrindo caminho para possíveis avanços em protocolos de comunicação segura.

Palavras-chave: Criptografia, Monoide de partição, Monoide de torre plana, Protocolo de Stickel, Sistemas de reescrita

Contents

List of Figures

1

Introduction

In a digital society, protecting information requires a combination of technical expertise and legal knowledge. Despite efforts, there is no guarantee of fulfilling all information security objectives.

Cryptography serves as the principal mechanism for implementing technical safeguards. The field of cryptography has witnessed significant evolution over time, notably in the advancement of two fundamental cryptographic techniques: symmetric and asymmetric encryption, also known as public-key encryption.

Key exchange protocols facilitate the secure establishment of shared keys between two parties, typically over an insecure communication channel. Often, these parties, referred to as Alice and Bob, generate a shared secret key without prior agreement to communicate via an open channel. The adversary's primary objective is to uncover this shared secret key, as its discovery compromises the encryption system.

In the past four decades, cryptographic protocols rooted in group and semigroup theory have garnered increasing attention and development.

Our research is driven by the exploration of algebraic structures like monoids within cryptographic key exchange protocols, aiming to assess their suitability as alternative platforms for Stickel's key exchange protocol. Consequently, our investigation is energised by the proposition of utilising the partition monoid as a new platform for Stickel's protocol, with a focus on obtaining normal forms as a necessary consideration. Recognising the importance of normal forms, we meticulously analyse a presentation for the partition monoid to determine its completeness, thus facilitating the acquisition of normal forms.

In the initial chapters, we introduce the fundamental concepts of elementary semigroup theory and essential principles for comprehending cryptographic aspects. We offer a succinct overview of the evolution of cryptography and explore topics such as public key encryption and various cryptographic protocols.

In the fourth chapter delves into an article by Myasnikov et al. (see [29]), focusing specifically on Stickel's key exchange protocol. Our investigation comprises a comprehensive examination of the protocol, which includes its implementation in Python and scrutiny of a linear algebra attack proposed by Myasnikov et al. Through exhaustive

testing, we experimentally corroborate Myasnikov's results.

In the fifth chapter further delves into a meticulous analysis of one of East's presentations for the partition monoid (see [8]), with the objective of assessing its completeness. Our investigation concludes that East's presentation does not constitute a complete presentation for the partition monoid. Lastly, in the sixth chapter, we shift our focus to a submonoid of the partition monoid, known as the planar rook monoid, and present a complete presentation for it.

The conclusion of our thesis contains the principal discoveries and contributions; and outlines potential directions for future research endeavours.

2

Preliminaries on Combinatorial Semigroup Theory

In this chapter, we will introduce some concepts and basic results that will be used throughout the thesis and we will review the fundamental principles of cryptography essential for understanding the cryptographic aspects.

2.1 Elementary semigroup theory

A *binary operation* $\circ$ on a set S is a map $\circ : S \times S \to S$. A set S equipped with a binary operation $\circ : S \times S \to S$ is a *semigroup* if the operation is *associative*, i.e., $x \circ (y \circ z) = (x \circ y) \circ z$ for all $x, y, z \in S$.

An element e in S serves as a *left identity* if $ex = x$ for all $x \in S$, a *right identity* if $xe = x$ for all $x \in S$, and as an *identity* if $ex = xe = x$ for all $x \in S$. If a semigroup has an identity, it is termed a *monoid*.

Consider an element z in S. If $zx = z$ for all $x \in S$, z is a *left zero*. If $xz = z$ for all $x \in S$, z is a *right zero*. When $zx = xz = z$ for all $x \in S$, z is a *two-sided zero* or simply a *zero*.

Suppose that we are given a semigroup S without an identity and introduce a new element 1 not in the semigroup S. Extend the multiplication on S to $S \cup \{1\}$ by defining $1x = x1 = x$ for all $x \in S$. This extension proves to be associative, rendering $S \cup \{1\}$ a monoid with identity 1. Similarly, with a new element 0 now in S, extend the multiplication on S to $S \cup \{0\}$ by $0x = x0 = 00 = 0$ for all $x \in S$. This extension maintains associativity, and $S \cup \{0\}$ is a semigroup with zero 0.

Let M be a monoid, and consider an element $x \in M$. If there exists x' such that $xx' = 1$, then x' is a *right inverse* for x, making x *right invertible*. Similarly, if there exists x'' such that $x''x = 1$, then x'' is a *left inverse* of x, and x is *left invertible*. If x is both left and right invertible, it is *invertible*.

Therefore, for an invertible element x in a monoid M, the unique right and left inverses of x are denoted by x^{-1}. A monoid where every element is invertible is termed a *group*.

Consider a non-empty subset T of S. It is a *subsemigroup* if it remains closed under

multiplication, i.e., $TT \subseteq T$. A *proper subsemigroup* refers to any subsemigroup excluding S itself. If a subsemigroup is also a monoid, it is referred to as a *submonoid*. If a subsemigroup is a group, it is termed a *subgroup*.

Let S be a semigroup and $\mathcal{T}_i = \{T_i : i \in T\}$ a collection of subsemigroups of S. If the intersection $\cap_{i \in I} T_i$ is non-empty, then it is also a subsemigroup.

Consider $X \subseteq S$, and let T be the collection of subsemigroups of S that contain X. Since S is one of such subsemigroups, T is non-empty, and its intersection is a subsemigroup. In fact, it is the smallest subsemigroup of S that contains X. The subsemigroup, denoted $\langle X \rangle$, is called the *subsemigroup generated by* X, and we have that $\langle X \rangle = \{x_1 x_2 \cdots x_n : n \in \mathbb{N}, x_i \in X\}$. If $\langle X \rangle = S$, then X is *generating set* for S, and X *generates* S. Therefore, if there exists a finite generating set for S, then S is *finitely generated*.

For a subset X of a monoid M, the submonoid generated by X, denoted $Mon\langle X \rangle$, is defined as the intersection of all submonoids of M that contain X and have 1_M as their identity. We can express $Mon\langle X \rangle$ as the set $\{1_M\} \cup \{x_1 x_2 \cdots x_n : n \in \mathbb{N} \cup \{0\}, x_i \in X\}$.

Let M be a monoid, as let X be a subset of M. If the submonoid generated by X, denoted as $Mon\langle X \rangle$, is equal to the entire monoid M, i.e. $Mon\langle X \rangle = M$, then X is a formally defined as a *monoid generating set* for M, and M is said to *generate* itself as a monoid.

It is worth noting that if X is a generating set for M, then X is also a monoid generating set. Conversely, if X is a monoid generating set for M, then $X \cup \{1_M\}$ is a generating set for M. Thus, M is finitely generated if and only if there is a finite monoid generating set for M.

For a subset X of a group G, the subgroup generated by X, denoted $Gp\langle X \rangle$, is defined as subgroups of G that contain X, have 1_G as their identity and their inverses via the group operation. We can express $Gp\langle X \rangle$ as the set $\{1_G\} \cup \{x_1 x_2 \cdots x_n : n \in \mathbb{N} \cup \{0\}, x_i \in X \text{ or } x_i^{-1} \in X\}$.

Let G be a group, as let X be a subset of G. If the subgroup generated by X, denoted as $Gp\langle X \rangle$, is equal to entire the group G, i.e. $Gp\langle X \rangle = G$, then X is defined as a *group generating set* for G.

2.2 Homomorphisms

Let S and T be semigroups. A map $\varphi : S \to T$ is a *homomorphism* if $(xy)\,\varphi = (x\varphi)\,(y\varphi)$ for all $x, y \in S$. Suppose now that S and T are monoids, then φ is a *monoid homomorphism* if $(xy)\,\varphi = (x\varphi)\,(y\varphi)$ for all $x, y \in S$ and $1_S\varphi = 1_T$.

A *monomorphism* is an injective homomorphism. A *surjective homomorphism* is also known as an epimorphism.

If $\varphi : S \to T$ is a surjective homomorphism, then T is a *homomorphic image* of S. An *isomorphism* is a bijective homomorphism. If there is an isomorphism $\varphi : S \to T$, then we say S and T are *isomorphic* and denote this by $S \simeq T$.

The *kernel* of homomorphism $\varphi : S \to T$ is the binary relation.

$$\ker \varphi = \{(x, y) \in S \times S : x\varphi = y\varphi\}.$$

2.3 Congruences and Quotients

A binary relation ρ on S is

- *left compatible* if $(\forall x, y, z \in S)\ (x\ \rho\ y \Rightarrow zx\ \rho\ zy)$;
- *right compatible* if $(\forall x, y, z \in S)\ (x\ \rho\ y \Rightarrow xz\ \rho\ yz)$;
- *compatible* if $(\forall x, y, z, t \in S)\ ((x\ \rho\ y) \wedge (z\ \rho\ t) \Rightarrow xz\ \rho\ yt)$.

A left compatible equivalence relation is a *left congruence*; a right compatible equivalence relation is a *right congruence*; and a compatible equivalence relation is a *congruence*.

Define ρ^C as the smallest left and right compatible relation containing ρ; and $\rho^\#$ is the smallest congruence containing ρ, called the *congruence generated* by ρ.

Consider a congruence ρ on S. The set of ρ-classes of S is the *quotient set* of S by ρ, denoted as S/ρ. Let $[x]_\rho \in S/\rho$ be the ρ-class of x for every $x \in S$. This may be expressed as follows: $[x]_\rho = \{y \in S : y\ \rho\ x\}$.

We define a multiplication on S/ρ by:

$$[x]_\rho\,[y]_\rho = [xy]_\rho\,.$$

This multiplication is well-defined, in the sense that if we chose different representatives for the ρ-classes $[x]_\rho$ and $[y]_\rho$, we would get the same answer:

$$([x] = [x']) \wedge ([y] = [y']) \Rightarrow (x\ \rho\ x') \wedge (y\ \rho\ y') \Rightarrow xy\ \rho\ x'y' \Rightarrow [xy]_\rho = [x'y']_\rho\,.$$

The set S/ρ, equipped with the defined multiplication, is commonly referred to as the *quotient* or *factor* of S by the equivalence relation ρ. The natural map $\rho^\# : S \to S/\rho$, which assigns each element x to its corresponding ρ-class $[x]_\rho$ is a surjective homomorphism.

Theorem 2.3.1. *(First Isomorphism Theorem)*
Let $\varphi : S \to T$ be a homomorphism. Then $\ker\varphi$ is a congruence, and $S/\ker\varphi \simeq im\varphi$.

2.4 Free semigroups

An *alphabet* is an abstract set of symbols called *letters*. Let A be an alphabet and let F be a semigroup. Let $\tau : A \to F$ be an embedding of A into F. If, for any semigroup S and map $\varphi : A \to S$, there is a unique homomorphism $\varphi^+ : F \to S$ such that $\tau\varphi = \varphi$, then the semigroup F is said to be *free* on A.

A *word* over A is a finite sequence $(a_1, a_2, \ldots, a_m)$, where each term a_i of the sequence is a letter from A. The *length* of this word is m. There is also word of length 0 is called the *empty word*.

The set of all words (including the empty word) over A is denoted A^* and the set of all non-empty words (that is, of length 1 or more) over A is denoted A^+.

The multiplication of sequences is defined by concatenation and this multiplication is associative:

$$\forall\,(a_1, a_2, \ldots, a_m)\,, (b_1, b_2, \ldots, b_n) \in A^*,$$

$$(a_1, a_2, \ldots, a_m)\,(b_1, b_2, \ldots, b_n) = (a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n)\,.$$

It can be seen that a semigroup is free on A if and only if it is isomorphic to A^+ (with the operation given by concatenation), and for this reason we will say that A^+ is *the* free semigroup on A.

Proposition 2.4.1. *Let A be an alphabet and let F be a semigroup. Then F is free on A if and only if $F \simeq A^+$*

A proof of 2.4.1 can be found in [4].

Similarly, the *free monoid* over a set can be seen as the monoid where its elements are all finite sequences of zero or more elements from that set. The operation in this monoid is defined by the concatenation of these elements, and the identity element is the unique sequence with zero elements, often called the empty word and denoted by ε or λ. When referring to the free monoid on a set A, we denote it by A^*.

We can write $a_1a_2\cdots a_n$ for $(a_1, a_2, \ldots, a_n)$ and ε for the empty word. We denote the length of $u \in A^*$ by $|u|$. Observe that $|u| = 0$ if, and only if $u = \varepsilon$.

2.5 Presentations

The concept of a semigroup presentation is to specify a semigroup as a quotient of a free semigroup: that is, as a quotient A^+/σ for some congruence σ on the free semigroup A^+. It is sufficient to specify the alphabet A in order to specify the free semigroup. To define the congruence σ, all that is needed is to define some binary relation ρ that produces σ.

The pair $\langle A|\rho\rangle$, where A is an alphabet and ρ is a binary relation on A^+, is denoted by *semigroup presentation*. We often call the elements of ρ (which are pairs words in A^+) *defining relations*. Thus, we think of semigroup presented by $\langle A|\rho\rangle$ as the largest semigroup generated by A and satisfying the defining relations in ρ. This presentation defines any semigroup isomorphic to $A^+/\rho^\#$. If A and ρ are finite, then a presentation is also finite. A semigroup is *finitely presented* if it can be defined by a finite presentation.

Let the alphabet A represent a generating set for S. Then, when S is given by a presentation $\langle A|\rho\rangle$, the words in A^+ act as *representing* elements of S. In general, each element of S will have more than one representative in A^+. If $u, v \in A^+$, $(u, v) \in \rho^\#$ represent the same element, that is, we say that u and v are *equal* in S, and we express them as follows: $u =_S v$.

An elementary ρ-transition is a pair $(u, v) \in \left(\rho^S\right)^C$, where ρ^S is the smallest symmetric relation containing ρ and called the *symmetric closure* of ρ. Thus, (u, v) is an elementary

ρ-transition if and only if v can be obtained from u by substituting a subword y for a subword x of u, where $(x, y) \in \rho$ or $(y, x) \in \rho$.

Proposition 2.5.1. *Let S be presented by $\langle A|\rho\rangle$. Then $u =_S v$ if and only if there is a sequence $u_0, \ldots, u_n$ with $u_0 = u$, $u_n = v$, and each (u_i, u_{i+1}) being an elementary ρ-transition.*

For more details, including the proof of this proposition, see [4].

We could go over freedom and presentations again, but with monoids rather than semigroups. A monoid F is a free on A if and only if $F \simeq A^*$. Each monoid is the quotient of a monoid that is free.

A *monoid presentation* is a pair $\langle A|\rho\rangle$ defining a monoid M; the generators A are monoid generators and the defining relations in ρ can be of the form (u, ε) or (ε, u). Also, $u =_M v$ if and only if there is a sequence of elementary transitions from u to v (the analogue of Proposition 2.5.1).

2.6 Normal forms

Any equivalence relation ρ on a set of objects M defines the quotient set M/ρ whose elements are equivalence classes. Description of the quotient set is referred to as the classification problem for M with respect to the equivalence relation. The *normal form* of an object M is a "selected representative" from the class $[M]$.

A normal form is required to possess two crucial properties:

1. Every object under consideration must have precisely one normal form.

2. Two objects sharing the same normal form must be equivalent.

The uniqueness requirement in property 1 is occasionally relaxed, permitting the normal form to be unique up to a simple equivalence.

Principal hiding mechanisms for cryptographic protocols often assume the existence of normal forms for group elements.

2.7 Rewriting systems

Rewriting is the process of substituting different words for subterms in a formula; this is sometimes made easier by reduction or rewriting systems. These systems are essentially made up of a set of objects, and relations that determine how these objects should be transformed. However, rewriting can be non-deterministic, allowing for the applicability of more than one rule or for a rule to be applied in many ways to a word. Rewriting systems give a variety of possible rule applications rather than a definite solution for converting one word into another.

In the presentation $\langle A|R\rangle$, the set R and its elements are referred to as *rewriting* system and *rewriting rules*, respectively. A rewriting rule $r \in R$ is often expressed as follows:

$r = (u, v)$ or alternatively expressed by $u \to_R v$. If R and A are both finite, we say that presentation $\langle A|R\rangle$ is *finite*.

A binary relation $\to_R$ on A^* is denoted as *single-step reduction* (when the context is clear we just use $\to$), as follows:

$$w_1 \to_R w_2 \Leftrightarrow w_1 = xuy \text{ and } w_2 = xvy$$

for any $x, y \in X^*$ and $(u, v) \in R$. The transitive and reflexive closure of $\to_R$ is denoted by $\xrightarrow{*}_R$. To represent the transitive closure of $\to_R$, we use $\xrightarrow{+}_R$. If there is a word $w_2 \in X^*$ such that $w_1 \to_R w_2$, then a word $w_1 \in X^*$ is said to be *R-reducible*. A term is referred to as *R-irreducible*, or just irreducible, if it is not R-reducible. The set of all R-irreducible words is denoted by $Irr\,(R)$.

The rewriting system R on A is considered *noetherian* if there are no infinite descending chains in the relation $\to_R$, in other words, the relation is well-founded.

$$w_1 \to_R w_2 \to_R w_3 \to_R \cdots \to_R w_N \to_R \ldots.$$

We say that a rewriting system is *confluent* if whenever we have $w_1 \xrightarrow{*}_R w_2$ and $w_1 \xrightarrow{*}_R w_2'$ there is a word $z \in A^*$ such that $w_2 \xrightarrow{*}_R z$ and $w_2' \xrightarrow{*}_R z$. If R is simultaneously noetherian and confluent we say that R is *complete*. A presentation is called noetherian, confluent, or complete if it possesses the respective properties in its rewriting system.

It is simple to check that, if R is a noetherian rewriting system, each congruence class of $M\,(A; R)$ contains at least one irreducible element. Assuming R is noetherian, then R is a complete rewriting system if and only if each congruence class of a complete rewriting system fixes a unique irreducible word in the class.

The following proposition is direct consequence of the First Isomorphism Theorem (Theorem 2.3.1).

Proposition 2.7.1. *Let M be a monoid, R a rewriting system on A and ρ the congruence generated by R. Let $\varphi : A^* \to M$ be a surjective homomorphism, such that*

1. *$\rho = \ker \varphi$; and*
2. *$\varphi\,(A)$ generates M.*

Then, the monoid M is defined by the presentation $\langle A|R\rangle$.

The following proposition provides us with sufficient conditions to ensure that a complete rewriting system exists for a monoid M.

Proposition 2.7.2. *Let M be a monoid and R a noetherian rewriting system on A. Let $\varphi : A^* \to M$ be a surjective homomorphism such that*

1. *$\varphi\,(u) = \varphi\,(v)$ for each relation $(u, v) \in R$;*
2. *the restriction of φ to $Irr\,(R)$ is one-to-one.*

Then, R is a complete rewriting system that defines M.

There are obvious analogous definitions and results to those above obtained by replacing monoid by semigroup and the free monoid A^* by the free semigroup A^+.

An application of Proposition 2.7.2 can be found in [23].

2.8 Background on Combinatorial Group Theory

In group theory, decision problems are important. These entail deciding whether a property holds for a certain input. For instance, we can be interested in deciding if two elements are conjugate or whether two subgroups are isomorphic. Variants of several decision problems involving subsets of groups have been studied more recently. For example, we may study the rational subset membership problem (a overview of it can be found in [21]). The conjugacy problem has also been approached in two distinct ways in [19] and [5]: either we want to determine if two elements are conjugate with a conjugator that belongs to a certain subset, or we want to determine whether an element has a conjugate in a certain subset. Similar variations have been studied for groups and semigroups for the intersection, twisted conjugacy, and equality problems (see more details in [1], [5] and [33]).

Algorithmic problems in (semi)group theory are classified into two types:

1. *Decision problems* are characterised by the following: when presented with a property $\mathcal{P}$ and an object $\mathcal{O}$, the task is to determine whether the object $\mathcal{O}$ has the property $\mathcal{P}$.

2. *Search problems*, on the other hand, are of the following nature: when provided with a property $\mathcal{P}$ and the knowledge that there are objects with the property $\mathcal{P}$, the objective is to identify as least one such object that possesses the property $\mathcal{P}$.

In group-based cryptography, it is typical for the security of a protocol to depend on a search version of a decision problem. This entails looking for a witness while being aware that a specific attribute holds for our input. These problems are mostly decidable, but their complexity is what attracts our interest. Group theorists have not studied search problems for a long time, but cryptography has piqued interest in them.

Some important algorithmic problems in groups were formulated by the German mathematician Max Dehn in the early 20th century. We will introduce some of Dehn's problems, such as the word problem and the conjugacy problem, as well as other relevant problems in cryptography. For further details, we refer the reader to [27].

2.8.1 The word problem

The *word problem* (WP) involves determining, given a recursive presentation of a group G and a element $g \in G$, whether or not g equals the identity element 1 in G. The word problem can be dissected into two distinct components, framed as list the "whether" and

"not" aspects, or equivalently, the "yes" and "no" parts. When a group is represented by a recursive presentation using generators and relators, the "yes" component of the word problem can be addressed through a recursive method.

Quoting the proposition of [29]:

Proposition 2.8.1. *Let $\langle X; R\rangle$ be a recursive presentation of a group G. Then the set all words $g \in G$ such that $g = 1$ in G is recursively enumerable.*

2.8.2 The conjugacy problem

The *conjugacy problem* (CP) involves determining, given a recursive presentation of a group G and two elements $g, h \in G$, whether or not there exists an element $x \in G$ such that $x^{-1}gx = h$. Similar to the word problem, the conjugacy problem can be subdivided into the "yes" and "no" parts. The "yes" part is always recursive, as one can systematically enumerate all the conjugates of a given element.

The *conjugacy search problem* (CSP) entails finding, given a recursive presentation of a group G and two conjugate elements $g, h \in G$, a specific element $x \in G$ such that $x^{-1}gx = h$. As mentioned earlier, the conjugacy search problem always has a recursive solution, allowing for the systematic listing of all conjugates associated with a given element. However, similar to the word search problem, this type of solution may be time-consuming and inefficient.

2.8.3 The decomposition and factorisation problems

The *decomposition problem* involves a recursive presentation of a group G, two subgroups $A, B \leq G$ and two elements $g, h \in G$. The task at hand is to determine whether there exist two elements $x \in A$ and $y \in B$ that satisfy $x \cdot g \cdot y = h$.

One of the natural ramifications of the conjugacy search problem is the *decomposition search problem*. This problem involves a recursive presentation of a group G, two subgroups A and B along with two elements $g, h \in G$. The objective is to identify two elements, $x \in A$ and $y \in B$, that satisfy $x \cdot g \cdot y = h$, assuming there exists at least one combination of x and y that satisfies this condition.

It is important to note that some x and y satisfying the equality $x \cdot g \cdot y = h$ always exist (e.g. $x = 1, y = g^{-1}h$), so the emphasis is on having them satisfy the conditions $x \in A$ and $y \in B$. Therefore, the term "subgroup-restricted" is typically not used for this decomposition search problem.

A special case of the decomposition search problem, where $A = B$, is known as the double coset problem. Another special case, where $g = 1$, deserves attention.

The *factorisation problem* deals with an element w in a recursively presented group G, along with two subgroups $A, B \leq G$. The aim is to ascertain whether there are two elements $a \in A$ and $b \in B$ such that $a \cdot b = w$.

The *factorisation search problem* involves an element w in a recursively presented group G and two recursively generated subgroups $A, B \leq G$. The objective is to find any two elements $a \in A$ and $b \in B$ that satisfy $a \cdot b = w$, provided at least one such pair of elements exists.

3

Cryptography

The concept of *information* is assumed to be well-understood, and a foundational comprehension of issues related to information security is essential for delving into cryptography. Information security aspects differ and adapt in response to specific situations and requirements. Historically, protocols and mechanisms have been developed to address security challenges in physical document-based information systems, often involving mathematical algorithms. Compliance with regulations and procedural methodologies are essential for ensuring security.

In a digital society, safeguarding information requires a combination of technical expertise and legal knowledge. Despite efforts, there is no complete guarantee of fulfilling every information security objective. Cryptography serves as the primary means to realise technical safeguards involving mathematical techniques related to confidentiality, data integrity, entity authentication, and data origin authentication. Four fundamental objectives, presented in [25], construct a framework for information security from all the information security objectives: (1) Privacy or Confidentiality; (2) Data integrity; (3) Authentication; and (4) Non-repudiation:

1. *Confidentiality* is a service that restricts access to information to only those authorised to have it. *Security* is a term that means the same as confidentiality and privacy.

2. *Data integrity* is a service that addresses the unauthorised changes to data. To ensure data integrity, one must have the ability to detect data manipulation by unauthorised parties, which includes actions such as insertion, deletion, and substitution.

3. *Authentication* is a service that deals with identification and applies to both individuals and information. In communication, it is essential that both parties identify each other, while information sent over a channel must be authenticated in terms of its origin, date of origin, data content, and time sent. Consequently, this aspect of cryptography is usually divided into two major categories: *entity authentication* and *data origin authentication*. Data origin authentication provides implicit data integrity by detecting any message modifications that alter the source.

4. *Non-repudiation* enables entities to prevent denial of their previous commitments or actions. In situations where disputes arise due to an entity denying certain actions, a trusted third party intervention is necessary to resolve the disagreement. For instance, an entity may grant authorisation for the purchase of property by another entity but later deny such authorisation.

The world of cryptography has undergone significant evolution over time, dating back to the era of Caesar and potentially even earlier. This evolution is evident in the development of two fundamental cryptographic techniques: symmetric and asymmetric encryption.

To begin with, there is a fundamental distinction between public key (or asymmetric), cryptographic techniques introduced in 1976, and symmetric ciphers, which have been in use since the time of Caesar or even earlier.

In symmetric cipher, having knowledge of the decryption key is essentially the same as, or often identical to, having knowledge of the encryption key. This means that two parties communicating with each other must establish an agreement on a shared secret before initiating communication over an open channel. In contrast, understanding the encryption and decryption keys in asymmetric ciphers does not involve the same information (through any feasible computation). For instance, the decryption key could be kept confidential, while the encryption key is made public, allowing multiple individuals to encrypt, but only one individual to decrypt.

To avoid ambiguity, a common convention is to use the term *private key* in association with public key cryptosystems and *secret key* in association with symmetric key cryptosystems. This convention is motivated by the following rationale: it takes two or more parties to *share* a secret, but a key is truly *private* only when one party alone knows it.

In [29], applications of algorithmic problems in group theory to cryptography are discussed, encompassing classical challenges like Dehn's problems (the word problem, the conjugacy problem, and the isomorphism problem) as well as new problems rooted in cryptography (see [32]).

The primary focus is on the foundational components of cryptography, specifically the methods that enable two entities, typically referred to as Alice and Bob, to create a *shared secret key* without prior agreement. These processes are *termed key establishment protocols*. It's essential to underscore that successfully establishing a shared secret key transitions Alice and Bob into the realm of symmetric cryptography. This approach offers notable benefits, as having a shared secret facilitates efficient encryption and decryption operations.

3.1 Public key encryption

Consider an entity, let's refer to it as Alice, possessing a public key denoted as e and a corresponding private key denoted as d. The real challenge will lie in attempting to

compute the private key d from the public key e (a task that, in secure systems, is nearly impossible).

The public key fundamentally guides the encryption process, while its private counterpart governs the decryption. When another entity, let's say Bob, intends to send a message to Alice, it uses Alice's public key to encrypt the message, resulting in a ciphertext. This ciphertext is then transmitted to Alice, who can decrypt it using her private key.

A crucial advantage of such systems is the ease of distributing authentic public keys, a task much simpler than securely sharing secret keys, as demanded by symmetric systems.

For an adversary, the primary objective is to decrypt the ciphertext intended for another entity. If they succeed, the encryption system is considered compromised. An even more significant achievement would be key recovery, implying the derivation of Alice's private key. If this occurs, the encryption system is declared completely compromised, as the adversary can now decrypt every ciphertext intended for Alice.

Consider an encryption scheme comprising sets of encryption and decryption transformations denoted as $\{E_e : e \in \mathcal{K}\}$ and $\{D_d : d \in \mathcal{K}\}$, respectively, where $\mathcal{K}$ represents the key space. The encryption scheme is termed *symmetric-key* if, for each associated encryption/decryption key pair (e, d), it is computationally "easy" to determine d knowing only e and to determine e from d. Since $e = d$ in the majority of practical symmetric key encryption schemes, the term symmetric key becomes appropriate.

3.1.1 From key establishment to encryption

Suppose that Alice and Bob share a secret key K. This key is a part of a set represented as $\mathcal{K}$. This set is typically referred to as *key space*.

Let $H : \mathcal{K} \to \{0,1\}^n$ be any (public) function from the set $\mathcal{K}$ to the set of bit strings of length n. It is recommended to select a sufficiently large value for n. For instance, if $\mathcal{K}$ is finite, n should be at the very least $log_2|\mathcal{K}|$. However, if $\mathcal{K}$ is infinite, n can be as large as computational resources allow. Such functions are commonly termed as *hash functions*. These hash functions often serve dual purposes: they act as condensed representations or digital imprints of data, and they also ensure the integrity of messages.

Encryption: Bob encrypts his message $m \in \{0,1\}^n$ as

$$E(m) = m \oplus H(K),$$

where $\oplus$ is addition modulo 2.

Decryption: Alice computes:

$$(m \oplus H(K)) \oplus H(K) = m \oplus (H(K) \oplus H(K)) = m,$$

thus recovering the message m.

Note that this encryption has an *expansion factor* of 1, meaning that the encryption of a message is of the same length as the original message.

3.2 Cryptographic protocols

A *protocol* is essentially a multi-participant process laid out as a series of instructions. These instructions detail the actions necessary for two or more entities to achieve a specific goal. In more detail, a key establishment protocol is a type of protocol through which a mutual secret is created and shared between two or more entities, paving the way for subsequent cryptographic endeavours, as pointed out in [25].

In [2], there is an introduction to a concise algebraic protocol designed for key establishment. This protocol aims to facilitate the exchange of secret key between two parties communicating only over an open channel. The strength of the protocol lies in the challenges associated with deciphering equations within algebraic constructs, especially within groups. As the protocol unfolds, each participant performs algebraic operations involving multiplications, followed by a transformation within a monoid or group.

After these calculations, their results are communicated via the open channel. Both entities then engage in a secondary calculation to obtain a mutual secret key. This post-calculation is based on an algorithm designed to solve the word problem in the monoid or group.

In the case that the protocol is group-based, [2] showed that an adversary (observing all communications over the public channel) can break the scheme and determine the secret key as long as a system of conjugate equations about the associated group is feasible to solve.

It is well documented that there are certain groups where the word problem is solvable in polynomial time but the conjugacy problem is hard (see, for example, [26]).

Now, we introduce some protocols, particularly the Diffie-Hellman key exchange protocol, because it serves as inspiration for the other protocols.

3.2.1 Protocols based on the conjugacy search problem

Consider a group G in which the word problem is solvable. For any elements $w, a \in G$, the notation w^a denotes $a^{-1}wa$. While the conjugacy decision problem is of great relevance in group theory, the conjugacy search problem carries more relevance in complexity theory, but is less enticing in the context of group theory. If it is established that u is conjugate to v, a possible method is to systematically compare words expressed as u^x with v until there is a match. This straightforward method, however, is often burdened with an exponential time complexity dependent on the length v making it largely inefficient.

Therefore, in the absence of any recognised alternative solutions for the *conjugacy search problem* within the group G, it is plausible to suggest that $x \to u^x$ is a one-way function and build a (public key) cryptography protocol on that:

Following a simple protocol, from [18]:

1. An element $w \in G$ is made public.

2. Alice selects a private element $a \in G$ and sends w^a to Bob.

3. Bob then selects his private element $b \in G$ and sends w^b to Alice.

4. Alice computes w^{ba}, while Bob calculates w^{ab}.

If a and b are selected from subgroups of a group of G where elements mutually commute, it follows that $ab = ba$. This guarantees that Alice and Bob can obtain a common private key: $w^{ab} = w^{ba}$. Public subgroups A and B within G are defined by their generating sets, ensuring that for any $a \in A$ and $b \in B$, the condition $ab = ba$ remains true.

Choosing an appropriate group structure for the given protocol is not straightforward. Some requirements for the group have been suggested in [31]:

($P0$) The group should be recognised and established. In particular, the conjugacy problem within the group should be either well studied or related to another known problem, possibly from another mathematical domain.

($P1$) The word problem in G should have a fast solution, ideally by a deterministic method in linear or quadratic time. Better yet, there should be a computationally efficient "normal form" for the elements of G.

($P2$) The conjugacy search problem should *not* be solvable in subexponential-time using a deterministic approach.

($P3$) The elements of G should be disguised in such as a way that x can not be retrieved from $x^{-1}wx$ by inspection.

($P4$) G should be a group of super-polynomial (i.e., exponential or "intermediate") growth. This implies that the number of elements of length n in G must grow faster than any polynomial in n. This is essential to prevent attacks that exploit the key space completely. The "length n" typically refers to the length of the shortest word representing a group element, but in a more general situation, it may refer to the length of a different description, such as, "information complexity".

3.2.2 Protocols based on the decomposition problem

In [29], it is showed that solving the conjugacy search problem is unnecessary for an adversary to get the common secret key the previous; it is sufficient to solve the seemingly easier decomposition search problem.

Note that the conjugacy search problem is a special case of the decomposition problem where w' is conjugate to w and $x = y^{-1}$. The claim that the decomposition problem should be easier than the conjugacy search problem is intuitively clear since it is generally easier to solve an equation with two unknowns than a special case of the same equation with just one unknown.

In [26], the authors give a formal description of a typical protocol based on the decomposition problem. There is a public group G, and two public subgroups $A, B \leq G$ commuting element-wise, i.e., $ab = ba$ for any $a \in A$, $b \in B$.

1. Alice randomly selects private elements $a_1a_2 \in A$. Then she sends the elements $u = a_1wa_2$ to Bob.

2. Bob randomly selects private elements $b_1b_2 \in B$. Then he sends the element $v = b_1wb_2$ to Alice.

3. Alice computes $K_A = a_1va_2 = a_1b_1wb_2a_2$, and Bob computes $K_B = b_1ub_2 = b_1a_1wb_2a_2$. Since $a_ib_i = b_ia_i$ in G, one has $K_A = K_B = K$ (as an element of G), which is now Alice's and Bob's common secret key.

3.2.3 The Diffie-Hellman Key Exchange Protocol

The domain of public key cryptography can be traced the seminal paper by Diffie and Hellman [6]. In 2002 [14], Martin Hellman also acknowledged Merkle's work: "The system (. . .) has since become known as Diffie-Hellman key exchange. Although this system was first described in an article by Diffie and myself, it is a public key distribution system, a concept developed by Merkle, and should therefore be called Diffie-Hellman-Merkle key exchange if names are to be associated with it. I hope this little pulpit can help in that effort to recognize Merkle contributed equally to the invention of public key cryptography". The algorithm was delineated and Diffie, Hellman, and Merkle recognised as its inventors in the now expired U. S. Patent 4,200,770.

We will now describe the protocol: group $\mathbb{Z}_p^*$ of integers modulo p, where p is prime and g is primitive mod p, is used in the simplest and original implementation of the protocol. A more general description of the protocol uses an arbitrary finite cyclic group.

1. Alice and Bob mutually settle on a finite cyclic group G and choose a generating element g in G. The group G will be written multiplicatively.

2. Alice selects a random natural number a and sends g^a to Bob.

3. Bob selects a random natural number b and sends g^b to Alice.

4. Alice computes $K_A = g^{ba}$.

5. Bob computes $K_B = g^{ab}$.

Since the operation is commutative, Alice and Bob both obtain a shared secret key, as $K = K_A = K_B$.

4

Stickel's Key Exchange Protocol

Stickel's protocol has similarities with the classical Diffie-Hellman protocol, although it cannot be considered a formal generalisation of the latter.

In 2005, Stickel proposed the following protocol (see [34]):

Let G be a public nonabelian finite group, and $a, b \in G$ be public elements such that $ab \neq ba$. Let N and M be the orders of a and b, respectively.

The process can be outlined as follows:

1. Alice picks two random natural numbers $n < N$, $m < M$ and sends $u = a^n b^m$ to Bob;
2. Bob picks two random natural numbers $r < N$, $s < M$ and sends $v = a^r b^s$ to Alice;
3. Alice computes $K_A = a^n v b^m = a^{n+r} b^{m+s}$;
4. Bob computes $K_B = a^r u b^s = a^{n+r} b^{m+s}$

Both Alice and Bob compute K, where $K = a^{n+r} b^{m+s}$, giving them a shared secret key.

There is also a variant of this protocol that seems favoured:

Let $w \in G$ be public.

1. Alice selects two random natural numbers $n < N, m < M$, an element c_1 from the center of the group G, and sends $u = c_1 a^n w b^m$ to Bob;
2. Bob selects two random natural numbers $r < N, s < M$, an element c_2 from the center of the group G, and sends $v = c_2 a^r w b^s$ to Alice;
3. Alice computes $K_A = c_1 a^n v b^m = c_1 c_2 a^{n+r} w b^{m+s}$;
4. Bob computes $K_B = c_2 a^r u b^s = c_1 c_2 a^{n+r} w b^{m+s}$.

It is worth noting that the elements c_i belong to the center of the group and so they commute with every other element of the group.

Although G has been described as a group, a semigroup would suffice. Stickel's suggestion is to use the group of invertible $k \times k$ matrices over a finite field F_{2^l}.

While this approach is adaptable to any semigroup, vulnerabilities are tied to the specific platform. Some attacks can be effective if G is a group, but might not apply universally across all semigroups.

Note: In the pursuit of acquiring the shared secret key K an adversary (Eve) only needs do identify any elements $x, y \in G$ such that

$$xa = ax,\ yb = by,\ u = xwy.$$

If Eve manages to identify such elements x and y, she can then leverage Bob's communicated value, $v = c_2 a^r w b^s$ to deduce:

$$xvy = xc_2a^r wb^s\ y = c_2a^r xwy\ b^s = c_2a^r ub^s = K.$$

It follows from this observation that multiplying by c_i does not really augment the protocol's security.

Solving the above system of equations in G is equivalent to solving the (subsemigroup-restricted) decomposition search problem:

Given a recursively presented (semi)group G, two recursively generated sub(semi)group groups $A, B \leq G$, and two elements $u, w \in G$, find two elements $x \in A$ and $y \in B$ that would satisfy $x \cdot w \cdot y = u$, provided that such a pair exists.

In the context of Stickel's scheme, the sub(semi)groups A and B are the centralisers of the elements a and b, respectively. This set is a subsemigroup of G; if G is group, then this set is a subgroup.

Stickel's scheme holds security equivalent to, or maybe weaker than, those models that are fundamentally built on the presumed difficulty of the decomposition search problem. This is because there exist strategies to compromise Stickel's scheme without attacking the relevant decomposition search problem. For instance, an approach proposed by Sramka [34] targets the retrieval of one of the exponents n, m, r, s in Stickel's protocol. Unlike other attacks that target only the shared secret key, Sramka's strategy aims to reveal a private key.

4.1 Linear algebra attack

In this section, we will present an attack to Stickel's key exchange protocol.

Consider G the group of invertible $k \times k$ matrices over a finite field $\mathbb{F}_{2^l}$, where $k = 31$. Recall that Stickel's proposal was to use this group with this value of k. Although the value of l was not specified in [34], it is reasonable to assume $2 \leq l \leq k$. The specific selection of matrices a, b, w is not so important for the attack; what is important is that a and b are invertible. We notice however that the choice of matrices a and b in [34] (the fact that the entries of these matrices are either 0 or 1) provides an extra weakness to the scheme.

The matrices stated in [34] are constructed as follows:

Let $p(x)$ and $q(x)$ be two different irreducible polynomials of degree n over the field $\mathbb{F}_2$ consisting of the zero and unit element only. Let a and b be the corresponding companion matrices. To be specific, if e.g.

$$p(x) = x^n + \Sigma_{i=0}^{n-1} c_i x^i$$

then

$$\begin{pmatrix} 0 & 0 & \cdots & \cdots & c_0 \\ 1 & 0 & 0 & \ddots & c_1 \\ 0 & 1 & \ddots & \ddots & c_2 \\ \vdots & \ddots & \ddots & 0 & c_{n-2} \\ 0 & 0 & \cdots & 1 & c_{n-1} \end{pmatrix}$$

is the corresponding $n \times n$ companion matrix.

Assume further that n, as well as, $2^n - 1$ are prime numbers. Such numbers $2^n - 1$ are called Mersenne primes and n is called Mersenne exponent.

Remember, that for Eve's success, it suffices to discover a solution to the system: $xa = ax, yb = by, u = xwy$, where a, b, u, w are known and x, y unknown $k \times k$ matrices over F_{2^l}. The first two equations can each be broken down into k^2 linear equations in terms of the matrix entries of x and y. A helpful manoeuvre is to multiply both sides of the equation $u = xwy$ by x^{-1} on the left to get

$$x^{-1}u = wy.$$

Given that $xa = ax$ is equivalent to $x^{-1}a = ax^{-1}$, let's use $x_1 = x^{-1}$ and replace the system of equations by:

$$x_1 a = a x_1,\ yb = by,\ x_1 u = wy.$$

Now, each equation here can again be decomposed into k^2 linear equations for the (unknown) entries of the matrices x_1 and y. The total number of linear equations to be addressed is $3k^2$ with $2k^2$ unknowns. Remember that a solution to this system helps to find the shared key K if and only if x_1 is invertible because $K = xvy$, where $x = x_1^{-1}$.

Using the known invertible matrix u, multiply both sides of the equation $x_1 u = wy$ by u^{-1} on the right to get $x_1 = wyu^{-1}$. By eliminating x_1 from the system, we obtain:

$$wyu^{-1}a = awyu^{-1}, yb = by.$$

Now, the only unknown is the matrix y, leading to $2k^2$ linear equations for k^2 entries of y.

It is worth noting that determining a matrix's invertibility is straightforward, as it aligns with matrix reduction to echelon form. Interestingly, in the tests of [29], there was often just a single variable left undefined, making the final step (checking for invertibility)

redundant. If there is a unique, non-zero solution to the system, then the associated matrix y must be invertible. In all our tests, as shown in the following section, the system has a unique non-zero solution, which means that we can experimentally corroborate the results of Myasnikov, Shpilrain and Ushakov.

One possible improvement to consider is the use of non-invertible components such as a, b, w. Specifically, this indicates that the foundational platform ought to be a semigroup containing non-invertible elements. If matrices are to be employed, it is logical to consider the semigroup of all $k \times k$ matrices over a finite ring (not necessarily a field). Such a semigroup typically has a lot of non-invertible elements. Hence, selecting a, b, w non-invertible should be feasible, rendering the linear algebra attack ineffective. Another benefit of not limiting to invertible matrices is the flexibility to employ not just powers a^j of a given element, but combinations like $\Sigma_{i=1}^{p} c_i \cdot a^i$, where c_i are constants from the foundational ring.

Stickel's scheme becomes compromised if the associated decomposition search problem is deciphered. Up to this point, no specific abstract (semi)group has been proved to resist current attack strategies targeting the decomposition search problem.

4.2 Linear algebra attack implementation

We wrote a code in Python implementing Stickel's key exchange protocol and constructing the linear algebra attack proposed by Myasnikov et al.

4.2.1 The code

Before explaining the code for the linear algebra attack described in the previous section, it should be noted that for the matrices proposed by Stickel, if $ab = ba$, then $a = b$ this will increase the efficiency of the code's implementation.

Proposition 4.2.1. *Let a, b be matrices over a finite field $\mathbb{F}_2$ constructed as proposed by Stickel (4.1). If $ab = ba$ then $a = b$.*

Proof. Let $a, b \in G$ be public elements of a public nonabelian finite group G. Assume that a and b are Stickel matrices of form:

$$a = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 & a_1 \\ 0 & 1 & 0 & 0 & \cdots & 0 & a_2 \\ 0 & 0 & 1 & 0 & \cdots & 0 & a_3 \\ 0 & 0 & 0 & 1 & \cdots & 0 & a_4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & a_k \end{bmatrix} \quad and \quad b = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 & b_1 \\ 0 & 1 & 0 & 0 & \cdots & 0 & b_2 \\ 0 & 0 & 1 & 0 & \cdots & 0 & b_3 \\ 0 & 0 & 0 & 1 & \cdots & 0 & b_4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & b_k \end{bmatrix}$$

Let the matrices product of ab and ba,

$$ab = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 1 & b_k \\ 0 & 0 & 0 & \cdots & 0 & a_1 & 1+a_1b_k \\ 1 & 0 & 0 & \cdots & 0 & a_2 & b_1+a_2b_k \\ 0 & 1 & 0 & \cdots & 0 & a_3 & b_2+a_3b_k \\ 0 & 0 & 1 & \cdots & 0 & a_4 & b_3+a_4b_k \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & a_k & b_{k-1}+a_kb_k \end{bmatrix}$$

and

$$ba = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 1 & a_k \\ 0 & 0 & 0 & \cdots & 0 & b_1 & 1+b_1a_k \\ 1 & 0 & 0 & \cdots & 0 & b_2 & a_1+b_2a_k \\ 0 & 1 & 0 & \cdots & 0 & b_3 & a_2+b_3a_k \\ 0 & 0 & 1 & \cdots & 0 & b_4 & a_3+b_4b_k \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & b_k & a_{k-1}+b_ka_k \end{bmatrix}$$

Assuming commutativity, we now have $ab = ba$, if and only if

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 1 & b_k \\ 0 & 0 & 0 & \cdots & 0 & a_1 & 1+a_1b_k \\ 1 & 0 & 0 & \cdots & 0 & a_2 & b_1+a_2b_k \\ 0 & 1 & 0 & \cdots & 0 & a_3 & b_2+a_3b_k \\ 0 & 0 & 1 & \cdots & 0 & a_4 & b_3+a_4b_k \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & a_k & b_{k-1}+a_kb_k \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 1 & a_k \\ 0 & 0 & 0 & \cdots & 0 & b_1 & 1+b_1a_k \\ 1 & 0 & 0 & \cdots & 0 & b_2 & a_1+b_2a_k \\ 0 & 1 & 0 & \cdots & 0 & b_3 & a_2+b_3a_k \\ 0 & 0 & 1 & \cdots & 0 & b_4 & a_3+b_4b_k \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & b_k & a_{k-1}+b_ka_k \end{bmatrix} \implies$$

$$\implies a = b$$

□

The code and comments can be found in Appendices A.

First, some auxiliary functions were created. These were used to perform operations between elements and between matrices.

The functions `f2add(a,b)` and `f2multiply(a,b)` are built to perform the sum and product in $\mathbb{F}_2$, respectively, not only of numbers but also variables.

The next auxiliary function built is `matrixaddF2` whose purpose is to add matrices with entries in $\mathbb{F}_2$. The function `matrixaddF2`, receives two square matrices (the entries can be numbers or variable), `a` and `b`, and the dimension of the matrices as input and uses the function `f2add` to calculate the sum of the matrices entry by entry in $\mathbb{F}_2$.

The function `matrixmultiplyF2` is designed to implement the product between matrices with entries in $\mathbb{F}_2$. Specifically, it uses the aforementioned `f2add` function for addition and `f2multiply` for multiplication.

Once the auxiliary functions are defined, the field is defined. To do this, we work with the `galois` module in Python. So we define field of order 2, `GF(2)`.

We defines two matrices, `a` and `b`, over $\mathbb{F}_2$ following Stickel's proposal: their subdiagonals are filled with $1's$ and their last columns are filled with the coefficients of the respective random irreducible polynomials of degree k constructed according to the constraint $a \neq b$. Then, we calculate the powers, a^n and b^m, and consequently $u = a^n b^m$ and its inverse.

For multiplication by a matrix Y of variables, the function `matmul()` does not work, so it is necessary to use the function we defined to perform the product of matrices in $\mathbb{F}_2$. It is necessary to make integer vector copies of matrices a^n, b^m and u^{-1} using the function `numpy.array`. These copies allow the calculation of products between matrices of elements and matrices of variables in the function `matrixmultiplyF2`. One improvement that could potentially be made is to avoid these copies, as this would make the code more efficient. For other products that do not involve variables, `matmul()` is used to make the code more efficient.

Once the matrix Y of symbols has been created, the two equations of the system are defined: `eq1` and `eq2`.

To solve the system, let's transform the problem into a system of the type $AX = 0$. This is done by moving the right-hand side (rhs) to the left-hand side (lhs), adding the two sides of equation. This results in a normalised equation (`eq1norm` and `eq2norm`), where the right-hand side is a zero matrix and the left-hand side is the sum of the two original sides. This is a sum, because in $\mathbb{F}_2$ addition and subtraction coincide.

The dimension of the null space, i.e. the number of basis vectors, indicates the number of free variables. The greater the dimension of the null space, the greater the number of free variables and the greater the number of solutions.

The code is working with a matrix equation in $\mathbb{F}_2$ and aims to find a solution matrix, `Ysolution`, that is invertible. If the null space has dimension 1, this means that there is only one basis vector, or in other words, one solution to the equation. If there is more than one possible solution due to a null space of dimension greater than 1, we try linear combinations of the solutions until we have an invertible matrix.

With the matrix Y calculated, it is possible to solve the equation $x_1 = yu^{-1}$, where $x_1 = x^{-1}$. Thus the solutions have been found.

The function `kfind(x,y,q)` was also defined with the purpose of computing the shared secret key, K, where the input are the previously discovered matrices, x and y, and a message, v, which Eve intercepts.

Finally, the function `kshared(a,b,r,s,m,n)` was generated, which is the "real" shared secret key, meaning that if the attack is successful, the matrix resulting from this function will be the same as the one discovered by Eve, `kfind(x,y,q)`.

4.2.2 Alice and Bob encryption

Now let's look at an example of Alice's and Bob's encryption and Eve's attack. Recall, $a, b \in G$ are public elements and, N and M are the orders of a and b, respectively. Suppose that $k = 7$:

1. Alice picks two random natural numbers $n < N$, $m < M$ and sends $u = a^n b^m$ to Bob;

```
Matrix a^(n):
[[1 1 0 0 0 0 1]
 [1 1 1 0 0 0 0]
 [1 1 1 1 0 0 0]
 [1 0 1 1 1 0 1]
 [1 0 0 1 1 1 1]
 [0 0 0 0 1 1 0]
 [1 0 0 0 0 1 1]]

Matrix b^(m):
[[1 0 0 0 1 0 0]
 [0 1 0 0 0 1 0]
 [0 0 1 0 0 0 1]
 [1 0 0 1 1 0 0]
 [0 1 0 0 1 1 0]
 [0 0 1 0 0 1 1]
 [0 0 0 1 0 0 1]]

Matrix u:
[[1 1 0 1 1 1 1]
 [1 1 1 0 1 1 1]
 [0 1 1 1 0 1 1]
 [0 1 1 0 1 1 0]
 [0 1 1 0 1 0 0]
 [0 1 1 0 1 0 1]
 [1 0 1 1 1 1 0]]
```

2. Bob picks two random natural numbers $r < N$, $s < M$ and sends $v = a^r b^s$ to Alice;

```
 Matrix a^(r):
[[0 0 0 1 0 1 0]
 [1 0 0 0 1 0 1]
 [0 1 0 0 0 1 0]
 [0 0 1 1 0 1 1]
 [1 0 0 0 1 1 1]
 [0 1 0 1 0 0 1]
 [0 0 1 0 1 0 0]]

Matrix b^(s):
[[1 1 1 1 1 1 0]
 [1 1 1 1 1 1 1]
 [0 1 1 1 1 1 1]
```

```
 [1 1 0 0 0 0 1]
 [1 1 1 0 0 0 0]
 [1 1 1 1 0 0 0]
 [1 1 1 1 1 0 0]]

Matrix v:
[[0 0 1 1 0 0 1]
 [1 1 1 0 0 1 0]
 [0 0 0 0 1 1 1]
 [1 0 1 1 0 1 0]
 [0 0 0 1 0 1 0]
 [1 1 0 0 0 1 0]
 [1 0 0 1 1 1 1]]
```

3. Alice computes $K_A = a^n v b^m = a^{n+r} b^{m+s}$;

4. Bob computes $K_B = a^r u b^s = a^{n+r} b^{m+s}$

```
Shared secret key matrix:
[[0 0 0 0 1 0 0]
 [0 0 0 1 1 0 0]
 [0 0 0 0 1 1 0]
 [0 0 0 1 0 0 1]
 [1 0 0 0 0 0 0]
 [0 1 0 1 0 1 0]
 [0 0 1 0 1 0 1]]
```

4.2.3 Eve's attack

Eve's goal is to discover the shared secret key $K = a^{n+r} b^{m+s}$. Recall that, as shown in Section 4, to obtain this key, it suffices to identify the elements $x, y \in G$ such that

$$\begin{cases} xa = ax \\ yb = by \\ u = xy, \end{cases}$$

where a, b, u are known.

After solving the system of equations, the solution `Ysolutionmatrix` represents exactly the element y that Eve needed to identify. Once y has been identified, the only task left is to find the element x. With $x^{-1} = yu^{-1}$, it possible to calculate the inverse of the element Eve needs to identify. Having identified x and y, Eve can proceed with protocol attack.

Eve's attack is made using the function defined by `kfind(x,y,q)`.

```
Matrix Y:
[[1 0 0 0 1 0 0]
 [0 1 0 0 0 1 0]
```

```
 [0 0 1 0 0 0 1]
 [1 0 0 1 1 0 0]
 [0 1 0 0 1 1 0]
 [0 0 1 0 0 1 1]
 [0 0 0 1 0 0 1]]

Matrix x^(-1)=x1:
[[1 1 0 1 1 0 1]
 [1 1 1 0 1 1 0]
 [0 1 1 1 0 1 1]
 [0 1 1 0 0 0 0]
 [0 1 1 0 1 0 1]
 [0 1 1 0 1 1 1]
 [1 0 1 1 0 1 1]]

Matrix x:
[[1 1 0 0 0 0 1]
 [1 1 1 0 0 0 0]
 [1 1 1 1 0 0 0]
 [1 0 1 1 1 0 1]
 [1 0 0 1 1 1 1]
 [0 0 0 0 1 1 0]
 [1 0 0 0 0 1 1]]

Secret key matrix (K):
[[0 0 0 0 1 0 0]
 [0 0 0 1 1 0 0]
 [0 0 0 0 1 1 0]
 [0 0 0 1 0 0 1]
 [1 0 0 0 0 0 0]
 [0 1 0 1 0 1 0]
 [0 0 1 0 1 0 1]]
```

4.3 Linear algebra attack analysis

The focus of our analysis lies in understanding the execution time of our algorithms for matrices of different sizes. The study of algorithmic behaviour with large inputs is commonly known as *asymptotic time complexity*.

The time complexity of an algorithm delineates the duration required for its execution in relation to the size of its input. There are "fast" algorithms, whose time complexity is given by some polynomial (or smaller than some polynomial), such as $O(logx)$, and "slow" algorithms, whose complexity is given by some exponential function.

A Turing machine is a "prototypical computer", a general abstract model of a "computing device". This device has a programme, reads and writes data from a tape and maintains an internal state. We used a machine to carry out the linear algebra attack for Stickel's protocol and perform time analysis, with the following layout:

antonio - 10.141.137.11
vnc://10.141.137.11
Hardware Overview:
Model Name: Mac Studio
Model Identifier: Mac13,2
Model Number: Z14K000YHPO/A
Chip: Apple M1 Ultra
Total Number of Cores: 20 (16 performance and 4 efficiency)
Memory: 128 GB
System Firmware Version: 8419.41.10
OS Loader Version: 8419.41.10
Serial Number (system): XHPF741LFD
Hardware UUID: 3AFA8CDC-E1DE-5229-9593-0E7B90C5F6B5
Provisioning UDID: 00006002-001C59242147401E
Activation Lock Status: Disabled

After successfully attacking the protocol, it is possible to draw some conclusions about the time it takes to break the protocol. Therefore, we decided to analyse how the time varies with k.

We collected 3 time counts for each value of k, resulting in an average of values for each k. The final data presents the average times (in seconds) from $k = 3$ to $k = 31$.

Remark 4.3.1. *Analysing the graph in Figure 4.1, the execution time of our algorithm appears to grow exponentially with the size of the matrices (k).*

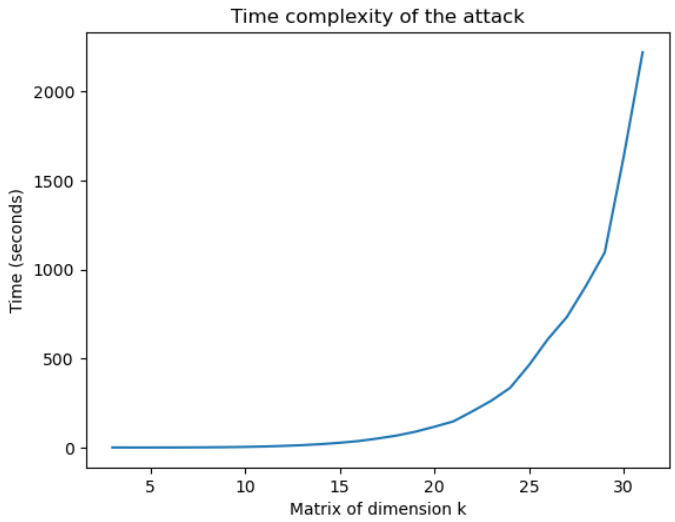


Figure 4.1: Time complexity attack: describes the amount of machine time it takes to execute an algorithm depending the dimension (k) of the matrix.

Moreover, in all our tests, there is only one non-trivial solution, allowing us to experimentally corroborate the results of Myasnikov et al. [29]. Notice that the matrix b^m will always be a non-trivial solution since m is less than the order of the matrix.

Suppose that $au^{-1} = u^{-1}a$.

In the system,

$$\begin{cases} yb = by \\ \\ yu^{-1}a = ayu^{-1} \end{cases} \iff \begin{cases} yb = by \\ \\ yau^{-1}u = ayu^{-1}u \end{cases} \iff \begin{cases} yb = by \\ \\ ya = ay. \end{cases}$$

It should be noted that Stickel [34] did not mention anything about the commutativity of a and b^{km}, for all k and the commutativity of b and any power of a. We are unsure if the fact that a and b^m commute implies a and b commute (and therefore $a = b$). Should this be the case, the protocol's definition a priori excludes them.

We do not know if this solution has not been presented to us because it is a priori excluded when we impose the condition that a and b not commute, or if there may exist the possibility of commutativity between a and b^m. In case it exists, b^{km} for all k becomes a non-trivial solution, just as a^{kr} for all k also becomes a non-trivial solution since r is less than the order of the matrix.

5

The partition monoid

Diagram algebras (i.e., algebras with a basis consisting of diagrams) have received considerable attention since the introduction of the Brauer algebra [3] in 1937. Other diagram algebras are the Temperly-Lieb algebra [11] and the Jones algebra [17]. The motivation for the study of these algebras is substantial, as they emerge in a diverse range of mathematical disciplines, including statistical mechanics, knot theory, and the investigation of algebraic groups. The previously mentioned three algebras, along with numerous more, are subalgebras of the partition algebras [24], which have as its basis all set-partitions of a $2n$-element set; these partitions are represented as (equivalence classes of) graphs on $2n$ vertices.

The study of Schur-Weyl duality in the representation theory of the symmetric group naturally leads to the partition algebra; a comprehensive exposition and a long list of references can be found in Halverson and Ram's survey-style article [13].

Diagram algebras can also be viewed as instances of semigroups, with examples including (full) transformation semigroups (see, for example, [15] or [16]) or the symmetric and dual symmetric inverse semigroups (see [20] and [10], respectively.)

Wilcox [35] actually realised the partition algebra as a twisted semigroup algebra of the partition monoid $\mathcal{P}_n$, which is also taken into consideration in [13]. From this perspective, a lot of information can be determined from different aspects of the monoid's structure, including the algebra's cellularity [12]. A characterisation of the semisimplicity of the partition algebras, the construction of Murphy elements and Specht modules, and a presentation of $\mathcal{P}_n$ in terms of generators and relations are among the main results of [13], although the proof provided for this presentation is far from being complete.

Such a presentation is crucial because it allows representations (homomorphisms into other algebras) to be built from presentations by selecting the images of the generators and checking that the relations are maintained. The multiplication of the partition algebra is complex, which makes this representation-building technique very desirable.

Let's consider the possibility of using the partition monoid as a platform for the Stickel's protocol. We begin by understanding the existence of a complete presentation in order to obtain normal forms. Therefore, we will study East's article and one of the

presentations he provides for the partition monoid to determine if it is complete or not.

In this chapter, we will present certain aspects of the structure of $\mathcal{P}_n$ and introduce three submonoids $\mathcal{L}_n, \mathcal{I}_n, \mathcal{R}_n$, where $\mathcal{L}_n$ is a submonoid of (an isomorphic copy of) the full transformation semigroup, $\mathcal{I}_n$ is (isomorphic to) the symmetric inverse semigroup, and $\mathcal{R}_n$ is an anti-isomorphic copy of $\mathcal{L}_n$. Also, we would like to highlight the existence of natural factorisation $\mathcal{P}_n = \mathcal{L}_n\mathcal{I}_n\mathcal{R}_n$. Next, we study East's presentation for $\mathcal{P}_n$ in which he uses this natural factorisation and the known presentations for the three submonoids. While the Halverson-Ram presentation has $3n - 2$ generators, this presentation has n^2 generators. Finally, we conclude that East's presentation is not a complete presentation for the partition monoid.

5.1 The partition monoid

Let $\mathbf{n}$ represent the finite set $\{1, \ldots, n\}$, where $\mathbf{n}$ is a positive integer that we will fix throughout this Section. Additionally, we define $\mathbf{n}' = \{1', \ldots, n'\}$ as a set corresponding to $\mathbf{n}$. A partition on $\mathbf{n} \cup \mathbf{n}'$, sometimes called just a partition, is a set of pairwise-disjoint non-empty sets whose union is $\mathbf{n} \cup \mathbf{n}'$. We will discuss an associative operation that creates a partition monoid, or set $\mathcal{P}_n$, containing all partitions of $\mathbf{n} \cup \mathbf{n}'$. The blocks are represented by sets $A_1, \ldots, A_k$ and we denote α as $\{A_1, \ldots, A_k\}$. A block A_i is said to be trivial if it has only one element. A graph on the vertex set $\mathbf{n} \cup \mathbf{n}'$ may represent a partition $\alpha \in \mathcal{P}_n$. Vertices $1, \ldots, n$ are arranged in a row (growing from left to right) and in a parallel row immediately below. After that, we add edges so that two vertices are connected by a path only if they are in the same α-block. To illustrate, the partition

$$\{\{1, 2', 3'\}, \{2\}, \{3\}, \{4, 1', 4', 5'\}, \{5, 6, 6'\}\} \in \mathcal{P}_6$$

is represented by the graph shown in figure 5.1.

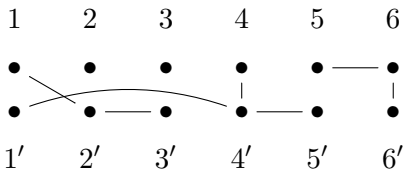


Figure 5.1: A graphical representation of a partition from $\in \mathcal{P}_6$.

While it is evident that such a graphical representation is not unique, we will identify two graphs on the vertex set $\mathbf{n} \cup \mathbf{n}'$ if they share the same connected components. We will make as distinction between a partition and a graph that represents it. To describe the product, let $\alpha, \beta \in \mathcal{P}_n$. Firstly, stack the graphs representing α and β so that vertices $1, \ldots, n$ of α are identified with vertices $1', \ldots, n'$ of β. Connect the edges of α with the edges of β and delete the middle row of vertices. The connected components of this graph are then constructed; the resulting graph is the product $\alpha\beta$. Figure 5.2 gives an example.

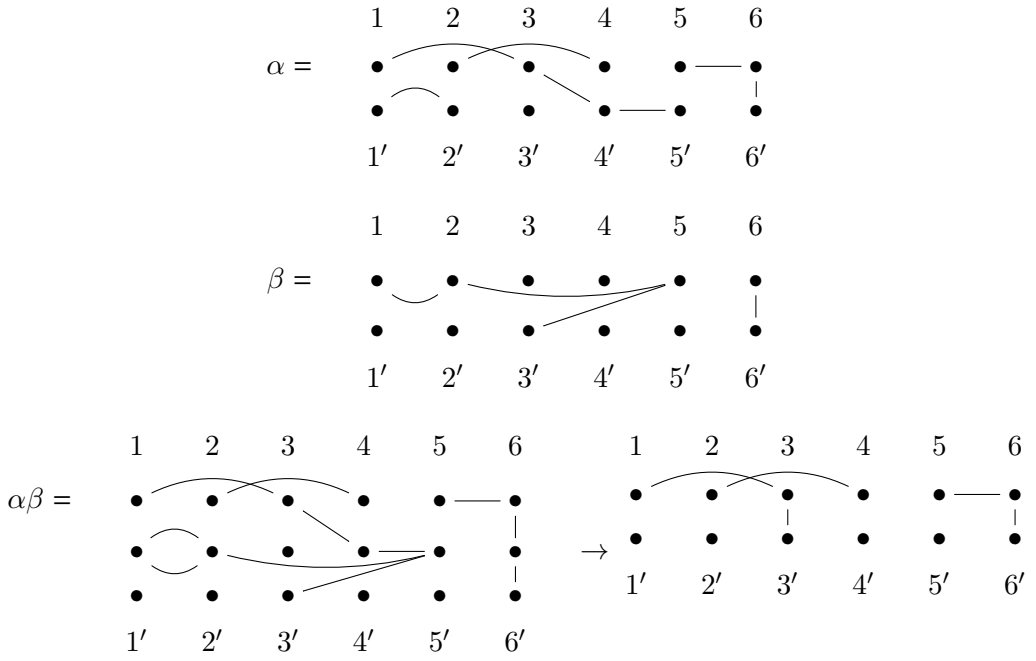


Figure 5.2: Calculating the product of two partitions $\alpha, \beta \in \mathcal{P}_6$.

We now go over the notation and terminology that we will be using to study partitions. Similar ideas from the theory of transformations semigroups serve as the basis for these definitions. In consideration of this, let $\alpha \in \mathcal{P}_n$. For $i \in \mathbf{n} \cup \mathbf{n}'$, let i_α represent the α block that contains i. The *domain* and *codamain* of α are defined as the sets

$$dom\,(\alpha) = \{i \in \mathbf{n} | i_\alpha \cap \mathbf{n}' \neq \emptyset\},$$

$$codom\,(\alpha) = \{i \in \mathbf{n} | i'_\alpha \cap \mathbf{n} \neq \emptyset\}.$$

Additionally, we define the *kernel* and the *cokernel* of α to be the equivalences

$$\ker(\alpha) = \{(i,j) \in \mathbf{n} \times \mathbf{n} | i_\alpha = j_\alpha\},$$

$$co\ker(\alpha) = \{(i,j) \in \mathbf{n} \times \mathbf{n} | i'_\alpha = j'_\alpha\}.$$

The $\ker(\alpha)$ and $co\ker(\alpha)$ equivalency classes of $\mathbf{n}$ are referred to as the *kernel-classes* and *cokernel-classes* of α. In order to demonstrate these concepts, let θ stand for the partition shown in figure 5.1. Then $dom\,(\theta) = \{1,4,5,6\}$ and $codom\,(\theta) = \{1,2,3,4,5,6\}$, and the *kernel-classes* and *cokernel-classes* are by $\{\{1\},\{2\},\{3\},\{4\},\{5,6\}\}$ and $\{\{1,4,5\},\{2,3\},\{6\}\}$, respectively.

The definitions make it clear right away that

$$dom\,(\alpha\beta) \subseteq dom\,(\alpha), \quad \ker(\alpha) \subseteq \ker(\alpha\beta),$$

$$codom\,(\alpha\beta) \subseteq codom\,(\beta), \quad co\ker(\beta) \subseteq co\ker(\alpha\beta)$$

for all $\alpha, \beta \in \mathcal{P}_n$.

5.2 Important subsemigroups

The partition monoid encompasses several well-researched and well-known subsemigroups, such as the (full) transformation semigroup and the symmetric and dual symmetric inverse semigroups. In this Section, our focus will be on the submonoid of the (full) transformation semigroup and the symmetric semigroup. For more details on these semigroups see [8].

5.2.1 The (full) transformation semigroup

The (regular) semigroup $\mathcal{T}_n$, encompassing all transformations on $\mathbf{n}$ or all functions from $\mathbf{n}$ to itself under the composition operation, is known as the (full) transformation semigroup on $\mathbf{n}$. The symmetric groups $\mathcal{S}_n$ is the group of units of $\mathcal{T}_n$ and consists of all permutations on $\mathbf{n}$. Let

$$\overline{\mathcal{T}_n} = \{\alpha \in \mathcal{P}_n | \textit{each block of } \alpha \textit{ contains exactly one element of } \mathbf{n}'\}$$

The graphical elements of $\overline{\mathcal{T}_n}$ reveals a natural one-one correspondence $\phi : \mathcal{T}_n \to \overline{\mathcal{T}_n}$. The surjective homomorphism $\phi : \mathcal{T}_n \to \overline{\mathcal{T}_n}$ is evident from the multiplication rules in $\mathcal{T}_n$ and $\mathcal{P}_n$, allowing us to identify the submonoid $\overline{\mathcal{T}_n}$ with the transformation semigroup $\mathcal{T}_n$. We often write $i\alpha$ for the image of i under the transformation associated with α, given $\alpha \in \overline{\mathcal{T}_n}$ and $i \in \mathbf{n}$.
The symmetric group $\mathcal{S}_n \subseteq \mathcal{T}_n$ is isomorphic to the subgroup $\overline{\mathcal{S}_n}$. The subgroup $\overline{\mathcal{S}_n}$ is identified as the group of units of $\overline{\mathcal{T}_n}$.

$$\overline{\mathcal{S}_n} = \{\alpha \in \mathcal{P}_n | \textit{each block of } \alpha \textit{ contains one element of } \mathbf{n} \textit{ and one of } \mathbf{n}'\},$$

In 1987, Moore [28] presented the symmetric group $\overline{\mathcal{S}_n}$, which we now describe.
For $1 \leqslant i \leqslant n-1$, we define $\overline{s_i} \in \overline{\mathcal{S}_n}$ to be simple transposition which interchanges i and i+1; see figure 5.3.

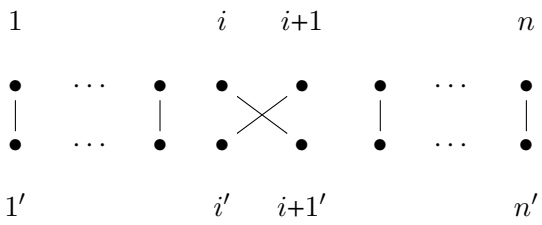


Figure 5.3: The simple transposition $\overline{s_i} \in \overline{\mathcal{S}_n}$.

Let $S = \{s_1, \ldots, s_{n-1}\}$ be alphabet, and consider the relations

(S1) $s_i^2 = 1$, for all i;
(S2) $s_i s_j = s_j s_i$, if $|i-j| > 1$;
(S3) $s_i s_j s_i = s_j s_i s_j$, if $|i-j| = 1$.

Theorem 5.2.1. *(Moore [28]) The symmetric group $\overline{\mathcal{S}_n}$ has (monoid) presentation $\langle S|\ (S1)-(S3)\rangle$ via*

$$\phi_S : S^* \to \overline{\mathcal{S}_n} : s_i \to \overline{s_i}.$$

The congruence on S^* generated by relations (S1)-(S3) is denoted by $\sim_S$.

5.2.1.1 The submonoid $\mathcal{L}_n$

Let $\alpha \in \overline{\mathcal{T}_n}$ and the *kernel-classes* of α be $K_1, \ldots, K_r$. For the set $\{min\ (K_1), \ldots, min\ (K_r)\}$, we write $Mins\ (\alpha)$. We say that α is block preserving if $i\alpha < j\alpha$ for every $i, j \in Mins\ (\alpha)$ and $i < j$. Put

$$\mathcal{L}_n = \{\alpha \in \overline{\mathcal{T}_n}|\ \alpha \textit{ is block-order-preserving and } codom\ (\alpha) = \mathbf{k} \textit{ for some } k \in \mathbf{n}'\}.$$

We will now describe a presentation for the submonoid $\mathcal{L}_n \subseteq \overline{\mathcal{T}_n}$.
In [7], it was demonstrated that this set is a submonoid of $\overline{\mathcal{T}_n}$. It was then examined in connection with the semigroup $\mathcal{T}_n \backslash \mathcal{S}_n$ of all non-invertible transformations on $\mathbf{n}$. Let $\overline{\lambda_{ij}} \in \mathcal{L}_n$, for $1 \leq i < j \leq n$ denote the map given by

$$x\overline{\lambda_{ij}} = \begin{cases} x & \text{if } 1 \leq x < j, \\ i & \text{if } x = j, \\ x-1 & \text{if } j < x \leq n. \end{cases}$$

This map is also shown in figure 5.4.

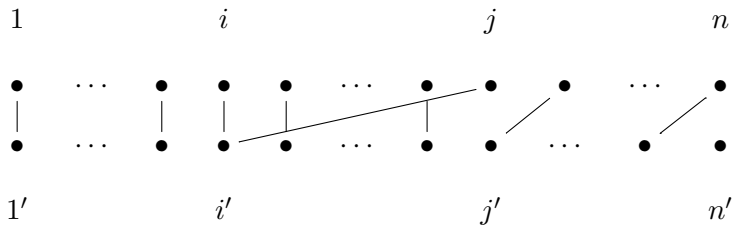


Figure 5.4: The map $\overline{\lambda_{ij}} \in \mathcal{L}_n$.

Define an alphabet $L = \{\lambda_{ij} | 1 \leq i < j \leq n\}$, and consider the relations

(L1) $\lambda_{kl}\lambda_{in} = \lambda_{kl}$, for all i, k, l;

(L2) $\lambda_{jk}\lambda_{ij} = \lambda_{ik}\lambda_{ij} = \lambda_{ij}\lambda_{i,k-1}$, if $i < j < k$;

(L3)-(L5) $\lambda_{kl}\lambda_{ij} = \begin{cases} \lambda_{ij}\lambda_{k-1,l-1}, & \text{if } i < j < k < l; \\ \lambda_{ij}\lambda_{k,l-1}, & \text{if } i < k < j < l; \\ \lambda_{i,j+1}\lambda_{kl}, & \text{if } i < k < l \leq j \leq n-1. \end{cases}$

Theorem 5.2.2. *(East [7]) The monoid $\mathcal{L}_n$ has presentation $\langle L|\ (L1)-(L5)\rangle$ via*

$$\phi_L : L^* \to \mathcal{L}_n : \lambda_{ij} \to \overline{\lambda_{ij}}.$$

The congruence on L^* generated by relations (L1)-(L5) is indicated by $\sim_L$, and for any $w \in L^*$, we write $\overline{w}$ for the transformation $w\phi_L \in \mathcal{L}_n$. Now, we will describe a set of normal forms for words over L. We define an order $\prec$ on the alphabet L by

$$\lambda_{ij} \prec \lambda_{rs} \text{ if and only if } i < r \text{ or } i = r \text{ and } j > s.$$

We define a word $\lambda_{i_1j_1} \dots \lambda_{i_kj_k} \in L^*$ to be ascending if $\lambda_{i_1j_1} \prec \cdots \prec \lambda_{i_kj_k}$. An ascending word $\lambda_{i_1j_1} \dots \lambda_{i_kj_k}$ is considered normal if $j_r \leq n - r{+}1$ for every r in $\mathbf{k}$. Let the set of all normal words over L be represented by $N_L \subseteq L^*$. Write $N_L^{(k)}$ for the set of all normal words of length k where $0 \leq k \leq n-1$. The following facts were proved in [7], as key steps in the proof of Theorem 5.2.2.

Lemma 5.2.3. *We have the following:*

1. *Every word over L is $\sim_L$-equivalent to a unique normal word.*
2. *If $w \in N_L^{(k)}$, then codom $(\overline{w}) = \{1, \dots, n-k\}$.*

Lemma 5.2.4. *We have the $\sim_L = \ker \phi_L$.*

The proof can be found in [7].

5.2.1.2 The submonoid $\mathcal{R}_n$

There exists a natural anti-involution $\wedge : \mathcal{P}_n \to \mathcal{P}_n$, mapping $\alpha \in \mathcal{P}_n$ to the partition $\hat{\alpha}$ obtained by reflecting α vertically. For example, if α a partition, then $\hat{\alpha}$ is obtained, as shown in figure 5.5.

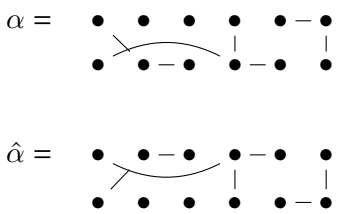


Figure 5.5: The diagrams of the elements α and $\hat{\alpha}$.

Let $\mathcal{R}_n = \hat{\mathcal{L}_n}$ represent the image monoid of $\mathcal{L}_n$ under the mapping; consequently, $\mathcal{R}_n$ is anti-isomorphic to $\mathcal{L}_n$. Based on the results of section 5.2.1.1, let $\overline{\rho_{ij}} \in \mathcal{R}_n$ for $1 \leq i < j \leq n$.

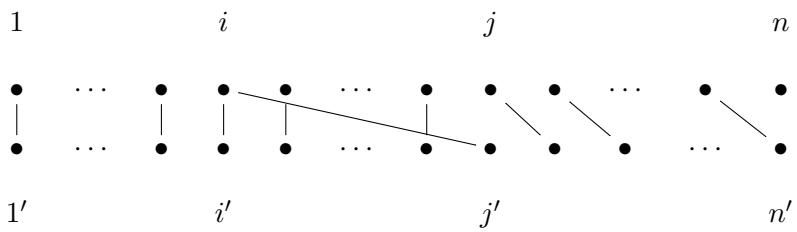


Figure 5.6: The map $\overline{\rho_{ij}} \in \mathcal{R}_n$.

It becomes apparent that $\mathcal{R}_n$ is generated by the partitions $\overline{\rho_{ij}}$. By inverting each word in the relations (L1)-(L5), it is possible to derive a presentation for $\mathcal{R}_n$ from the one for $\mathcal{L}_n$ outlined in 5.2.2. Define an alphabet $R = \{\rho_{ij} | 1 \leq i < j \leq n\}$, and consider the relations

(R1) $\rho_{in}\rho_{kl} = \rho_{kl}$ for all i, k, l;

(R2) $\rho_{ij}\rho_{jk} = \rho_{ij}\rho_{ik} = \rho_{i,k-1}\rho_{ij}$ if $i < j < k$;

(R3)-(R5) $$\rho_{ij}\rho_{kl} = \begin{cases} \rho_{k-1,l-1}\rho_{ij}, & \text{if } i < j < k < l; \\ \rho_{k,l-1}\rho_{ij}, & \text{if } i < k < j < l; \\ \rho_{kl}\rho_{i,j+1}, & \text{if } i < k < l \leq j \leq n-1. \end{cases}$$

Theorem 5.2.5. *The monoid $\mathcal{R}_n$ has a presentation $\langle R | (R1) - (R5) \rangle$ via*

$$\phi_R : R^* \to \mathcal{R}_n : \rho_{ij} \to \overline{\rho_{ij}}.$$

The congruence on R^* generated by relations (R1)-(R5) is indicated by $\sim_R$, and for any $w \in R^*$, we write $\overline{w}$ for the partitions $w\phi_R \in \mathcal{R}_n$. We may convert knowledge about $\mathcal{L}_n$ into information about $\mathcal{R}_n$ (and vice-versa) because to the duality between relations (L1)-(L5) and (R1)-(R5). Formally, we define an anti-isomorphism as:

$$\wedge : L^* \to R^* : \lambda_{ij} \to \rho_{ij}.$$

Following this, we have $w_1 \sim_L w_2$ if and only if $\hat{w}_1 \sim_R \hat{w}_2$. Remember that the set of all normal words over L is denoted by N_L. The elements of the set $N_R = \hat{N_L} = \{\hat{w} | w \in N_L\}$ are now defined as normal words over R. The set of all normal words (over R) of length k is denoted as $N_R^{(k)}$.

Lemma 5.2.6. *We have the following:*

1. *Every word over R is $\sim_R$-equivalent to a unique normal word.*
2. *If $w \in N_R^{(k)}$, then dom $(\overline{w}) = \{1, \ldots, n-k\}$.*

5.2.2 The symmetric inverse semigroup

5.2.2.1 A presentation for submonoid $\mathcal{I}_n$

The partial transformation on **n** is the (regular) semigroup $\mathcal{PT}_n$ of all partial transformations on **n**, or all partially defined functions from **n** to itself.
A natural injective map $\phi' : \mathcal{PT}_n \to \mathcal{P}_n$ exists. If $n \geq 2$, this map is not a homomorphism: let α and β be elements of $\mathcal{PT}_2$ shown in figure 5.7. Then, the multiplication rule in $\mathcal{PT}_2$ yields the *empty* partial transformation represented below by $\alpha\beta$ and multiplying in $\mathcal{P}_2$ returns $(\alpha\phi')\,(\beta\phi')$, as illustrated in figure 5.8.

$\alpha =$  $\beta =$ 

$\alpha\beta =$

Figure 5.7: The elements α and β of $\mathcal{PT}_2$.

$(\alpha\phi')\,(\beta\phi') =$

Figure 5.8: The product of α and β in $\mathcal{P}_2$.

Restricting ϕ' to the symmetric inverse semigroup $\mathcal{I}_n \subseteq \mathcal{PT}_n$, which is the semigroup of all injective partial transformations on $\mathbf{n}$, makes it simple to verify that we indeed obtain an embedding. As a result, we can associate $\mathcal{I}_n$ with its image

$$\overline{\mathcal{I}_n} = \{\alpha \in \overline{\mathcal{T}_n} | \textit{every non-trivial block of } \alpha \textit{ contains exactly one element of } \mathbf{n} \textit{ and one of } \mathbf{n}'\}.$$

under ϕ'. In figure 5.9, an example of an element of $\mathcal{I}_6$ is shown.

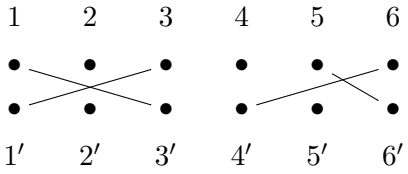


Figure 5.9: An element of $\mathcal{I}_6$.

Now, we will look at Popova's presentation [30] for $\mathcal{I}_n$, which expands upon Moore's presentation for $\mathcal{S}_n$. Let $\overline{\varepsilon} \in \mathcal{I}_n$ be the partial permutation, given by:

$\overline{\varepsilon} =$

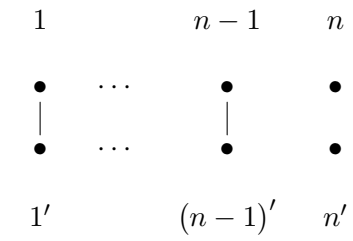


Figure 5.10: The map $\overline{\varepsilon} \in \mathcal{I}_n$.

Given the alphabet $I = S \cup \{\varepsilon\}$, consider the relations

(I1) $s_i^2 = 1$, for all i;
(I2) $s_i s_j = s_j s_i$, if $|i - j| > 1$;
(I3) $s_i s_j s_i = s_j s_i s_j$, if $|i - j| = 1$;
(I4) $\varepsilon^2 = \varepsilon$;
(I5) $\varepsilon s_i = \varepsilon s_i$, if $i \leq n - 2$;
(I6) $\varepsilon s_{n-1} \varepsilon s_{n-1} = s_{n-1} \varepsilon s_{n-1} \varepsilon = \varepsilon s_{n-1} \varepsilon$.

Theorem 5.2.7. *(Popova [30]) The symmetric inverse semigroup $\mathcal{I}_n$ has presentation $\langle I \mid (I1) - (I6)\rangle$ via*

$$\phi_I : I^* \to \mathcal{I}_n = \begin{cases} s_i \to \overline{s_i}, \\ \\ \varepsilon \to \overline{\varepsilon}. \end{cases}$$

The congruence on I^* generated by relations (I1)-(I6) is indicated by $\sim_I$. Once more, for $w \in I^*$, we denote by $\overline{w}$ the partial permutation $w\phi_I \in \mathcal{I}_n$.
For each $1 \leq i \leq n$, we define the word as $\varepsilon_i = (s_i \dots s_{n-1})\, \varepsilon\, (s_{n-1} \dots s_i)$ and for every $1 \leq i \leq n$+1, we have $e_i = \varepsilon_n \dots \varepsilon_i$. The maps are represented in figures 5.11 and 5.12, respectively.

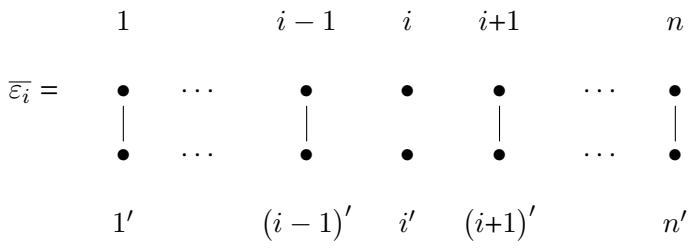


Figure 5.11: The map $\overline{\varepsilon_i} \in \mathcal{I}_n$.

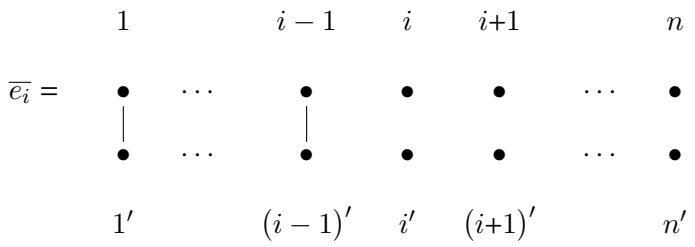


Figure 5.12: The map $\overline{e_i} \in \mathcal{I}_n$.

We define the rewriting system on the generating set I, orienting the relations (I1)-(I6) from left to right since it appears to be the "natural" orientation for performing reductions. Thus, the rewriting system (I1)-(I6) is defined as follows:

(I1) $s_i^2 \to 1$, for all i;
(I2) $s_j s_i \to s_i s_j$, if $j - i > 1$;
(I3) $s_j s_i s_j \to s_i s_j s_i$, if $j = i$+1;
(I4) $\varepsilon^2 \to \varepsilon$;
(I5) $\varepsilon s_i \to s_i \varepsilon$, if $i \leq n - 2$;
(I6) $\varepsilon s_{n-1} \varepsilon s_{n-1} \to s_{n-1} \varepsilon s_{n-1} \varepsilon \to \varepsilon s_{n-1} \varepsilon$.

The Shortlex is a total ordering for finite sequences of objects that can themselves be totally ordered. In the Shortlex order. The words are sorted by length of the words and when this first criteria do not specified the order, .i.e. when two words have the same length, then we use the criteria of the lexicographical order.
The lexicographic order on I is $\varepsilon > s_{n-1} > s_{n-2} > \dots s_j > \dots > s_i > \dots > s_1$, where $i < j$.

Lemma 5.2.8. *The rewriting system (I1)-(I6) is noetherian.*

Proof. We define a total order $\prec$ on the alphabet I by

$$s_i \prec s_j \prec \varepsilon \quad \text{if and only if } i < j < n.$$

Next, it is necessary guarantee that there are no infinite descending sequences.
We consider I^* with a Shortlex order. Let $u, v \in I^*$ and if $u \to v$, then we have $u > v$.
Thus, we can conclude the rewriting system is noetherian. □

Proposition 5.2.9. *The rewriting system (I1)-(I6) is not confluent and so not complete.*

Proof. Consider the word $s_{n-1}\varepsilon s_{n-1}\varepsilon s_{n-2}$. We aim reduce it as follows:

$$s_{n-1}\varepsilon s_{n-1}\varepsilon s_{n-2} \xrightarrow{*} \varepsilon s_{n-1}s_{n-2}\varepsilon.$$

But the same word can still be reduced as,

$$s_{n-1}\varepsilon s_{n-1}\varepsilon s_{n-2} \xrightarrow{*} s_{n-1}\varepsilon s_{n-1}s_{n-2}\varepsilon.$$

Since both words obtained after reduction are irreducible, it is not possible to find a common word w such that $\varepsilon s_{n-1}s_{n-2}\varepsilon \xrightarrow{*} w$ and $s_{n-1}\varepsilon s_{n-2}\varepsilon \xrightarrow{*} w$. The rewriting system (I1)-(I6) is not confluent and so not complete.

□

The next result shows that there is a natural factorisation $\mathcal{P}_n = \mathcal{L}_n\mathcal{I}_n\mathcal{R}_n$ and it serves as the foundation for how we derive a presentation for $\mathcal{P}_n$.

Proposition 5.2.10. *Let $\alpha \in \mathcal{P}_n$. Then $\alpha = \beta\gamma\delta$ for unique $\beta \in \mathcal{L}_n$, $\gamma \in \mathcal{I}_n$ and $\delta \in \mathcal{R}_n$, with $dom(\gamma) \subseteq codom(\beta)$ and $codom(\gamma) \subseteq dom(\delta)$.*

The proof of this proposition can be found in [8].

5.3 A presentation for the partition monoid

We now introduce a presentation for the partition monoid $\mathcal{P}_n$ using the results from the previous subsection. The surjective homomorphism is defined using Proposition 5.2.10.

$$\phi : (L \cup I \cup R)^* \to \mathcal{P}_n : \begin{cases} \lambda_{ij} \mapsto \overline{\lambda}_{ij}, \\ s_r \mapsto \overline{s}_r, \\ \varepsilon \mapsto \overline{\varepsilon}, \\ \rho_{ij} \mapsto \overline{\rho}_{ij}. \end{cases}$$

Consider the relations

(RL1)-(RL9) $\rho_{kl}\lambda_{ij} = \begin{cases} \varepsilon\lambda_{i-1,j-1}\rho_{kl}, & \text{if } l < i; \\ \varepsilon\lambda_{k,j-1}\rho_{kl}, & \text{if } l = i; \\ \varepsilon\lambda_{i,j-1}\rho_{kl}, & \text{if } i < l < j; \\ \varepsilon\lambda_{ki}\rho_{ki}, & \text{if } k < i < j = l; \\ \varepsilon, & \text{if } i = k < j = l; \\ \varepsilon\lambda_{ik}\rho_{ik}, & \text{if } i < k < l = j; \\ \varepsilon\lambda_{ij}\rho_{k,l-1}, & \text{if } k < j < l; \\ \varepsilon\lambda_{ij}\rho_{i,l-1}, & \text{if } j = k; \\ \varepsilon\lambda_{ij}\rho_{k-1,l-1}, & \text{if } j < k; \end{cases}$

(SL1)-(SL8) $s_r\lambda_{ij} = \begin{cases} \lambda_{ij}s_{r-1}, & \text{if } r > j; \\ \lambda_{i,j+1}, & \text{if } r = j; \\ \lambda_{i,j-1}, & \text{if } r = j-1 > i; \\ \lambda_{ij}, & \text{if } r = j-1 = i; \\ \lambda_{ij}s_r, & \text{if } i < r < j-1; \\ \lambda_{i+1,j}s_r, & \text{if } i = r < j-1; \\ \lambda_{i-1,j}s_r, & \text{if } r = i-1; \\ \lambda_{ij}s_r, & \text{if } r < i-1; \end{cases}$

(RS1)-(RS8) $\rho_{ij}s_r = \begin{cases} s_{r-1}\rho_{ij}, & \text{if } r > j; \\ \rho_{i,j+1}, & \text{if } r = j; \\ \rho_{i,j-1}, & \text{if } r = j-1 > i; \\ \rho_{ij}, & \text{if } r = j-1 = i; \\ s_r\rho_{ij}, & \text{if } i < r < j-1; \\ s_r\rho_{i+1,j}, & \text{if } i = r < j-1; \\ s_r\rho_{i-1,j}, & \text{if } r = i-1; \\ s_r\rho_{ij}, & \text{if } r < i-1; \end{cases}$

(EL1) $\lambda_{ij}\varepsilon = \lambda_{ij}$, for all i, j;

(EL2) $\varepsilon\lambda_{\text{in}} = \varepsilon$, for all i;

(EL3) $\varepsilon\lambda_{ij} = \lambda_{ij}s_{n-1}\varepsilon$, if $j < n$;

(RE1) $\varepsilon\rho_{ij} = \rho_{ij}$, for all i, j;

(RE2) $\varepsilon = \varepsilon$, for all i;

(RE3) $\rho_{ij}\varepsilon = \varepsilon s_{n-1}\rho_{ij}$, if $j < n$.

Symbolically, represent by $\sim$ the congruence on $(L \cup I \cup R)^*$ generated by the relations (L1)-(L5), (R1)-(R5), (I1)-(I6), (RL1)-(RL9), (SL1)-(SL8), (RS1)-(RS8), (EL1)-(EL3), (RE1)-(RE3). Thus, $\mathcal{P}_n$ has the following presentation via ϕ:

$$\left\langle L \cup I \cup R \mid \begin{array}{c} \text{(L1)-(L5), (R1)-(R5), (I1)-(I6), (RL1)-(RL9),} \\ \text{(SL1)-(SL8), (RS1)-(RS8), (EL1)-(EL3), (RE1)-(RE3).} \end{array} \right\rangle$$

Following our usual overline notation, write $\overline{w} = w\phi \in \mathcal{P}_n$ for each word w in $(L \cup I \cup R)^*$.

Proposition 5.3.1. *The East's presentation is not complete presentation for the partition monoid.*

Proof. For a rewriting system to define a complete presentation, it is necessary for the rewriting system to satisfy two properties: being confluent and noetherian.
The same word we used in proposition 5.2.9 continues to derive $\varepsilon s_{n-1}s_{n-2}\varepsilon \xrightarrow{*} w$ and $s_{n-1}\varepsilon s_{n-2}\varepsilon \xrightarrow{*} w$; and these words are irreducible. Therefore, the same argument is valid to prove that the rewriting system proposed by East for the partition monoid does not satisfy the confluence property.
In conclusion, East's presentation does not define a complete presentation for the partition monoid.

□

In this section, our goal was to verify if the presentation given by East was complete. We concluded that the rewriting system does not define a complete presentation for the partition monoid.
We therefore decided to study a submonoid of the partition monoid, the planar rook monoid, with the aim of providing a complete presentation for it. The importance of having a complete presentation lies in the existence of normal forms, since they will be necessary to consider the partition monoid as a platform for the Stickel's key exchange protocol. The following section is dedicated to this endeavour.

6

The planar rook monoid

6.1 The planar rook monoid

The planar rook monoid, denoted $\mathcal{PR}_n$, is the submonoid of the rook monoid $\mathcal{I}_n$ whose graphical representations are planar, meaning they have no crossings edges. Note that in the rook monoid the blocks are identified with the edges in the graphical representation. In the graphical representation of an element of $\mathcal{PR}_n$, we can easily list the elements in the domain, which are in the top line of the diagram, and their corresponding images in the bottom line of the diagram.
In the figures 6.1 and 6.2, we can observe some examples of elements of the planar rook monoid and non-elements, respectively, using a diagramatic form.

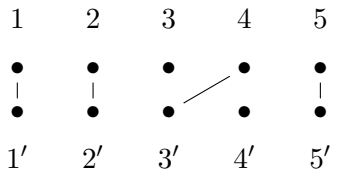


Figure 6.1: An element of the planar rook monoid.

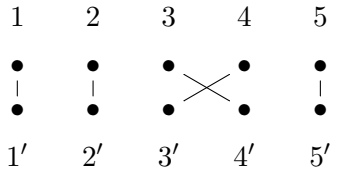


Figure 6.2: A non-element of the planar rook monoid

The elements of the planar rook monoid can also be viewed as order preserving one-to-one maps on the ordered set $\{1 < 2 < \cdots < n\}$. A map α is said to be order preserving if, for all $a, b \in Dom(\alpha)$, $a\alpha \leq b\alpha$, whenever $a < b$. The requirement for our maps to be one-to-one order preserving ensures that our diagrams are planar.
The monoid of all order preserving one-to-one maps on a chain with n elements is also known as the monoid $\mathcal{POI}_n$ of all injective order preserving partial transformations (see [9]).

We shall refer to the elements of the planar rook monoid as planar diagrams. The *rank* of a planar diagram in $\mathcal{PR}_n$ is the number of edges of its graphical representation.
The planar rook monoid has size $\binom{2n}{n}$ as we are choosing a subset of size n from a set of size $2n$. Indeed, a subset X of $\mathbf{n} \cup \mathbf{n}'$ of size n determines a planar diagram α: take $X \cap \mathbf{n}$ as the domain of α, and $X \cap \mathbf{n}'$ as $\mathbf{n}' \backslash codom\,(\alpha)$. In figure 6.1, the set $X = \{1, 2, 4, 5, 4'\}$ determines the given planar diagram.

6.2 The generators

In this section, we will exhibit a set of generators for $\mathcal{PR}_n$.
Consider the following elements of the Planar Rook Monoid: For $i \in \{1, \ldots, n-1\}$, let $\overline{r_i}$ denote the map given by

$$x\overline{r_i} = \begin{cases} x' & \text{if } 1 \leq x < i \ \wedge \ i{+}1 < x \leq n, \\ (x{+}1)' & \text{if } x = i, \end{cases}$$

whose diagram is shown in the following figure:

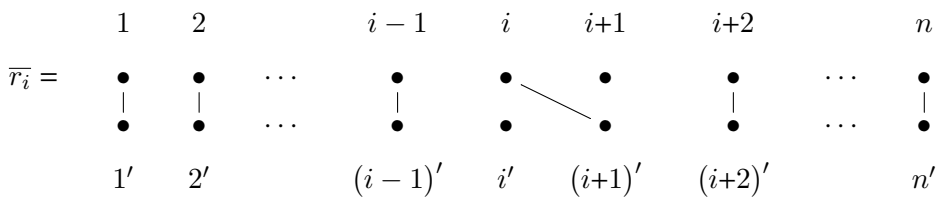


Figure 6.3: The map $\overline{r_i} \in \mathcal{PR}_n$

For $i \in \{1, \ldots, n-1\}$, let $\overline{l_i}$ denote the map by

$$x\overline{l_i} = \begin{cases} x' & \text{if } 1 \leq x < i \ \wedge \ i{+}1 < x \leq n, \\ (x-1)' & \text{if } x = i{+}1, \end{cases}$$

whose diagram is shown in the following figure:

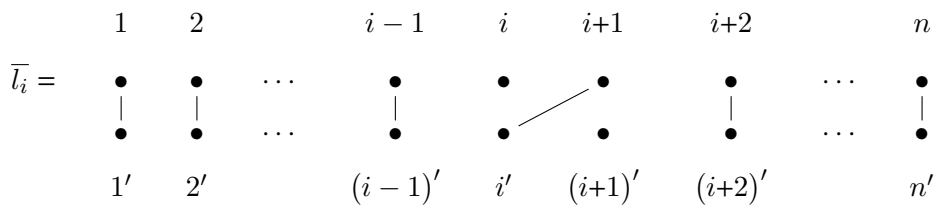


Figure 6.4: The map $\overline{l_i} \in \mathcal{PR}_n$

For $i \in \{1, \ldots, n\}$, let $\overline{e_i}$ denote the map by

$$x\overline{e_i} = x \qquad \text{if } 1 \leq x < i \ \wedge \ i{+}1 \leq x \leq n,$$

whose diagram is shown in the following figure:

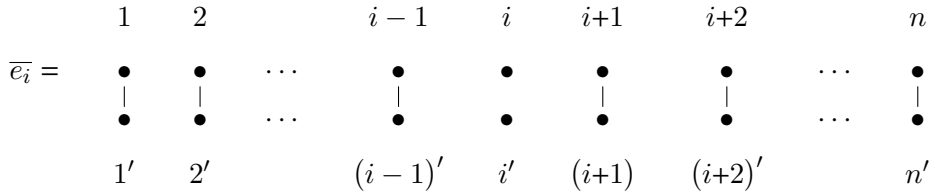


Figure 6.5: The map $\overline{e_i} \in \mathcal{PR}_n$

Let A be the alphabet $A = \{r_1, \ldots, r_{n-1}\} \cup \{l_1, \ldots, l_{n-1}\} \cup \{e_1, \ldots, e_n\}$. Let ϕ be the homomorphism given by

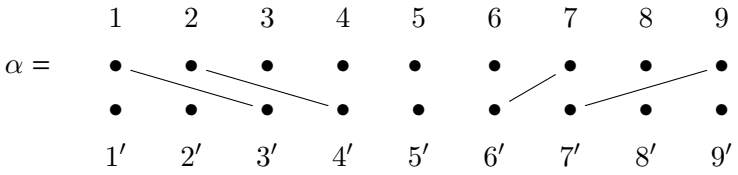
$$\phi : A^* \to \mathcal{PR}_n : \begin{cases} r_i \mapsto \overline{r_i}, \\ l_i \mapsto \overline{l_i}, \\ e_i \mapsto \overline{e_i}. \end{cases}$$

The goal of this subsection is to prove the following result:

Proposition 6.2.1. *$A\phi$ generates $\mathcal{PR}_n$.*

We begin with an example.

Example 6.2.2. *Consider the following element α of $\mathcal{PR}_n$ given by the diagram*



We have $rank\,(\alpha) = 4$, $dom\,(\alpha) = \{1, 2, 7, 9\}$ and $codom\,(\alpha) = \{3', 4', 6', 7'\}$. It is easy to check that $\alpha = \overline{r_2 r_3}\,\overline{r_1 r_2}\,\overline{e_5}\,\overline{l_6}\,\overline{l_8 l_7}$ as it can be seen in the following figure: .

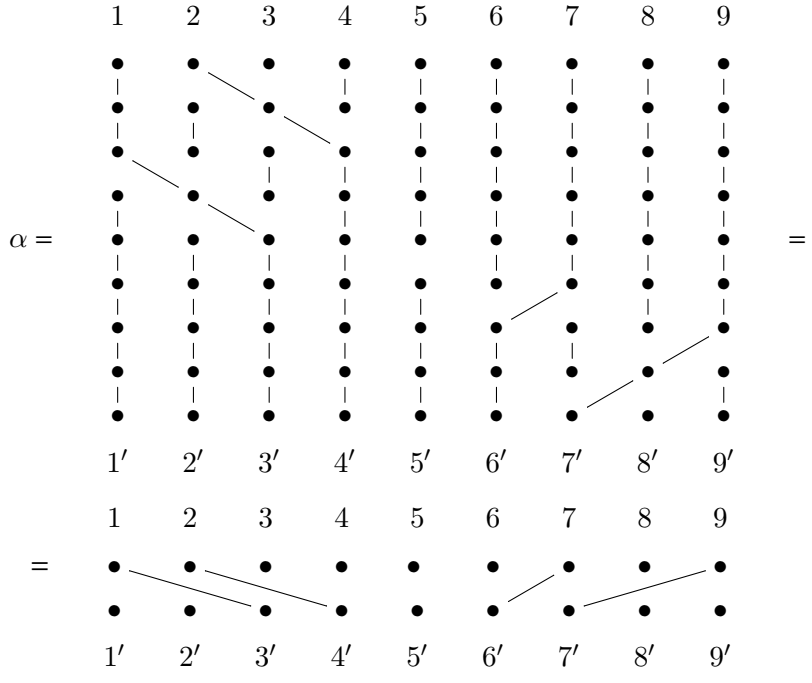
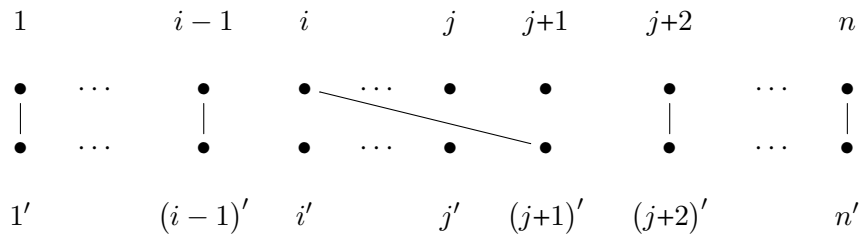


Figure 6.6: The map $\overline{r_2 r_3 r_1 r_2 e_5 l_6 l_8 l_7} \in \mathcal{PR}_n$

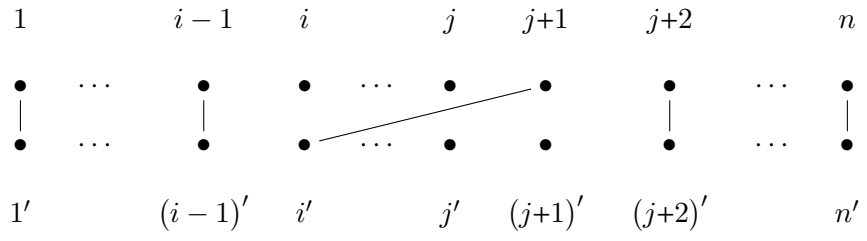
For $1 \leq i \leq j < n$, let $\overline{r_{ij}}$ denote the element of $\mathcal{PR}_n$ given by $\overline{r_i}\,\overline{r_{i+1}} \dots \overline{r_{j-1}}\,\overline{r_j}$.
The following lemma follows easily:

Lemma 6.2.3. *The element $\overline{r_{ij}}$ is the map of $\mathcal{PR}_n$ whose graphical representation is given by*



For $1 \leq i \leq j < n$, let $\overline{l_{ij}}$ denote the element of $\mathcal{PR}_n$ given by $\overline{l_j}\,\overline{l_{j-1}} \dots \overline{l_{i+1}}\,\overline{l_i}$.

Lemma 6.2.4. *The element $\overline{l_{ij}}$ is the map of $\mathcal{PR}_n$ whose graphical representation is given by*



Definition 6.2.5. *Given $i_1, \dots, i_k, j_1, \dots, j_k \in \mathbf{n}$, with $k \in \{1, \dots, n-1\}$, we say that the pair of sequences $((i_1, \dots, i_k), (j_1, \dots, j_k))$ satisfies the r-block condition if:*

1. $i_1 < \dots < i_k$;
2. $j_1 < \dots < j_k$;
3. $i_s < j_s$, *for all* $1 \leq s \leq k$; *and*
4. $i_{s+1} \leq j_s$+1, *for all* $1 \leq s < k$.

Definition 6.2.6. *We say that an element $\alpha \in \mathcal{PR}_n$ has an r-block if there exists $k \in \{1, \dots, n-1\}$ and blocks $\{i_s, j_s'\}$, for $1 \leq s \leq k$, with $((i_1, \dots, i_k), (j_1, \dots, j_k))$ satisfying the r-block condition. An r-block of α is said to be maximal if it is not contained in any other r-block of α.*

Example 6.2.7 (Continued 6.2.2)**.** *The element α has 4 non-trivial blocks, namely $\{1, 3'\}, \{2, 4'\}, \{6', 7\}, \{7', 9\}$. Also, α has a maximal r-block given by the set $\{1, 3'\} \cup \{2, 4'\}$.*

Similarly we can define l-blocks.

Definition 6.2.8. *Given $i_1, \dots, i_k, j_1, \dots, j_k \in \mathbf{n}$, with $k \in \{1, \dots, n-1\}$, we say that the pair of sequences $((i_1, \dots, i_k), (j_1, \dots, j_k))$ satisfies the l-block condition if:*

1. $i_1 < \cdots < i_k$;
2. $j_1 < \cdots < j_k$;
3. $j_s < i_s$, *for all* $1 \leq s \leq k$; *and*
4. $j_{s+1} \leq i_s$+1, *for all* $1 \leq s < k$.

Definition 6.2.9. *We say that an element $\alpha \in \mathcal{PR}_n$ has an l-block if there exists $k \in \{1, \ldots, n-1\}$ and blocks $\{i_s, j'_s\}$, for $1 \leq s \leq k$, with $((i_1, \ldots, i_k), (j_1, \ldots, j_k))$ satisfying the l-block condition. An l-block of α is said to be maximal if it is not contained in any other l-block of α.*

Example 6.2.10 (Continued 6.2.2)**.** *The set $\{6', 7\} \cup \{7', 9\}$ is the only maximal l-block of α.*

Lemma 6.2.11. *The element of $\mathcal{PR}_n$, $\overline{r_{i_k j_k}} \ldots \overline{r_{i_1 j_1}}$, with $((i_1, \ldots, i_k), (j_1, \ldots, j_k))$ satisfying the r-block condition, has a unique maximal r-block $\{i_1, j'_1\} \cup \cdots \cup \{i_k, j'_k\}$.*

Lemma 6.2.12. *The element of $\mathcal{PR}_n$, $\overline{l_{i_1 j_1}} \ldots \overline{l_{i_k j_k}}$, with $((i_1, \ldots, i_k), (j_1, \ldots, j_k))$ satisfying the l-block condition, has a unique maximal l-block $\{i_1, j'_1\} \cup \cdots \cup \{i_k, j'_k\}$.*

Definition 6.2.13. *An interval on $\mathbf{n}$ is any subset of the form $\{i, i+1, \ldots, j-1, j\}$, for $i, j \in \mathbf{n}$, with $i \leq j$, which we denote by $[i, j]$. Similarly, we define intervals in $\mathbf{n}'$, which we denote by $[i', j']$, for $i', j' \in \mathbf{n}'$, with $i' \leq j'$.*

Lemma 6.2.14. *For any $\alpha \in \mathcal{PR}_n$, a maximal r-block (respectively, l-block) is contained in a minimal (with respect to inclusion) interval $I = [i, j] \cup [i', j']$, for fixed $i, j \in \mathbf{n}$, in such a way that any non-trivial block in α does not intersect the interval I.*

Proof. Consider a maximal r-block $\{i_1, j'_1\} \cup \cdots \cup \{i_k, j'_k\}$ of α as in Definition 6.2.6. Take $i = i_1$ and $j = j_k$+1. Clearly, the maximal r-block $\{i_1, j'_1\} \cup \cdots \cup \{i_k, j'_k\}$ is contained in $I = [i, j] \cup [i', j']$. It follows from the maximally of the r-block that a non-trivial block of α as a trivial intersection with I.
The proof is similarly for the l-block case. □

Definition 6.2.15. *The index of a maximal r-block (respectively, maximal l-block) in an element α is the interval defined in Lemma 6.2.14.*

Definition 6.2.16. *We shall call a simple block in an element $\alpha \in \mathcal{PR}_n$ any block of the form $\{i, i'\}$, for fixed $i \in \mathbf{n}$.*

Lemma 6.2.17. *Any element α in $\mathcal{PR}_n$, has a unique decomposition, as union of blocks, into maximal r-blocks, maximal l-blocks, simple blocks and trivial blocks.*

Proof. Non trivial blocks in α have the form $\{i, j'\}$ for $i, j \in \mathbf{n}$, which can be simple blocks, r-blocks or l-blocks. Each maximal r-block is a union of r-blocks. Similarly, for maximal l-block. So, α can be viewed as the union simple blocks, maximal r-blocks, maximal l-blocks and trivial blocks. The uniqueness of the decomposition follows from Lemma 6.2.14. □

For $1 \leq i \leq j < n$, let r_{ij} denote the element of A^* given by $r_i\, r_{i+1} \ldots r_{j-1}\, r_j$ and l_{ij} denote the element $l_j\, l_{j\text{-}1} \ldots l_{i1}\, l_i$ of A^*. Applying the homomorphism ϕ, the image of r_{ij} is given by $\overline{r_{ij}}$ and the image of l_{ij}, is given by $\overline{l_{ij}}$ i.e., $r_{ij}\phi = \overline{r_{ij}}$ and $l_{ij}\phi = \overline{l_{ij}}$.
We are now able to prove the main proposition of this subsection.

Proof of Proposition 6.2.1. Let $\alpha \in \mathcal{PR}_n$. By Lemma 6.2.17, α can be viewed has a union of maximal r-blocks, maximal l-blocks, simple blocks and some trivial blocks.
By Lemma 6.2.14, any maximal r-block and any maximal l-block is contained in a minimal interval $[i, j] \cup [i', j']$. For each maximal r-block in α, let R_{ij} denote the element $\overline{r_{i_k j_k}} \ldots \overline{r_{i_1 j_1}}$, with $i = i_1$ and $j = j_k$+1, as in Lemma 6.2.11. Similarly, let L_{ij} denote the element $\overline{l_{i_1 j_1}} \ldots \overline{l_{i_k j_k}}$, with $i = j_1$ and $j = i_k$+1, as in Lemma 6.2.12.
Denote by X the set of indexes from the set $\mathbf{n}$, which are not the index of a maximal r-block of α, nor in the index of a maximal l-block of α, nor in a simple block of α.
The element α is the product of the elements R_{ij}, L_{ij}, corresponding to the maximal r-blocks and l-blocks, respectively, and the e_t's, with $t \in X$.
We can conclude that elements $\overline{r_i}$, $\overline{l_i}$ and $\overline{e_i}$ generate all elements of $\mathcal{PR}_n$, thus completing the proof. □

Lemma 6.2.18. *Any element of $\mathcal{PR}_n$ can be uniquely written as the product of maximal r-blocks, maximal l-block, and e_i's, in such a way that the indexes are in ascending order.*

Proof. Note that if R_{ij} (or L_{ij}) and R_{st}(or L_{st}) correspond to a maximal r-block (or a maximal l-block), then $[i, j\text{+}1] \cap [s, t\text{+}1] = \emptyset$, and so $R_{ij}R_{st} = R_{st}R_{ij}$ (or $L_{ij}L_{st} = L_{st}L_{ij}$). Also, if R_{ij} (or L_{ij}) corresponds to a maximal r-block (or l-block) and $k \in X$ with X as in the proof of Proposition 6.5, then $[i, j\text{+}1] \cap \{k\} = \emptyset$, and so $R_{ij}e_k = e_k R_{ij}$ (or $L_{ij}e_k = e_k L_{ij}$). □

Example 6.2.19. *(Continued 6.2.2)*
As we identified previously, the element α has 4 non-trivial blocks, namely $\{1, 3'\}, \{2, 4'\}, \{6', 7\}, \{7', 9\}$. The maximal r-block is given by the set $\{1, 3'\} \cup \{2, 4'\}$, and according to the Definition 6.2.5 and Lemma 6.2.11, we can express the maximal r-block as follows: $\overline{r_{23}}\,\overline{r_{12}}$.
The maximal l-block = $\{6', 7\} \cup \{7', 9\}$, and according to the Definition 6.2.8 and Lemma 6.2.12, we can express the maximal l-block as follows: $\overline{l_6}\,\overline{l_{78}}$.
Also, we identify the trivial blocks: $\{5\}$ and $\{5'\}$, which we can represent with the element $\overline{e_5}$.
By Lemma 6.2.17, we can represent any element in $\mathcal{PR}_n$ by the product of the maximal r-blocks, maximal l-blocks, simple blocks and trivial blocks. Thus, with the blocks identified, we used the Lemma 6.2.18 to write the element.
As the indexes are in ascending order, we write the element as $\overline{r_{23}}\,\overline{r_{12}}\;\overline{e_5}\;\overline{l_6}\,\overline{l_{78}}$, or it can also be written as $\overline{r_2 r_3}\,\overline{r_1 r_2}\;\overline{e_5}\;\overline{l_6}\,\overline{l_8 l_7}$.

Let N be the subset of A^* where words have the form

$$w_{\left[s_1, s_1^*\right]} \cdots w_{\left[s_t, s_t^*\right]},$$

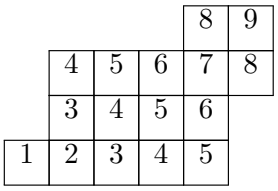
with $s_1 \leq s_1^* < s_2 \leq s_2^* < \cdots < s_t \leq s_t^*$ and $s_l, s_l^* \in \mathbf{n}$, where each $w_{\left[s_l, s_l^*\right]}$ can be either:

1. $r_{i_k j_k} \ldots r_{i_1 j_1}$, with $\left((i_1, \ldots, i_k), (j_1, \ldots, j_k)\right)$ satisfying the r-block condition, and $s_l = i_1$ and $s_l^* = j_k$+1; or
2. $l_{i_1 j_1} \ldots l_{i_k j_k}$, with $\left((i_1, \ldots, i_k), (j_1, \ldots, j_k)\right)$ satisfying the l-block condition, and $s_l = j_1$ and $s_l^* = i_k$+1; or
3. e_i with $s_l = s_l^* = i$.

Furthermore, if $w_{\left[s_l, s_l^*\right]}$ is of types 1 or 2 and $w_{\left[s_{l+1}, s_{l+1}^*\right]}$ is of type 3, then s_l^*+1 $< s_{l+1}$.

6.2.0.1 Combinatorial representation of r-blocks

In this subsection we will introduce a combinatorial representation of the r-blocks previously mentioned. This new representation will allow us to clearly understand how can we have another reading of the r-blocks.
By Lemma 6.2.11, the element of $\mathcal{PR}_n$, $\overline{r_{i_k j_k}} \ldots \overline{r_{i_1 j_1}}$, with $\left((i_1, \ldots, i_k), (j_1, \ldots, j_k)\right)$ satisfying the r-block condition, has a unique maximal r-block $\{i_1, j_1'\} \cup \cdots \cup \{i_k, j_k'\}$.
Let $\left((i_1, \ldots, i_k), (j_1, \ldots, j_k)\right)$ be a pair of sequences satisfying the r-block condition. For each $s \in \{1, \ldots, k\}$, consider a row sequence of $j_s - i_s$+1 boxes filled with the numbers from i_s up to j_s, which we will call the s-row. Now, for $s \in \{1, \ldots, k-1\}$ position the s+1-row in top of the s-row in such a way that the a box filled with a symbol $t \in \mathbf{n}$ is in top of a box filled with the symbol $t - 1$. We will obtain a diagram of boxes filled with symbols from $\mathbf{n}$, which we call the r-block diagram of the pair of sequences.
For example, the pair $\left((1, 3, 4, 8), (5, 6, 8, 9)\right)$ has r-block diagram:



Note that a similar l-block diagram could be drawn from a pair of sequences $\left((i_1, \ldots, i_k), (j_1, \ldots, j_k)\right)$ satisfying the l-block condition.
We claim that the r-block diagrams obtained from sequences satisfying the r-block condition have a skew shape. Indeed, condition (1) of Definition 6.2.5, implies that the s+1-row has leftmost box either in the north or in the northeast of the leftmost box of the s-row; condition (2), implies a similar condition but for the rightmost box; and condition (4), ensures that the leftmost box of the s+1-row is to the north or northwest of the rightmost box of the s-row.
Also note that an l-block diagram also has a skew shape.
We shall call the row reading of an r-block diagram to the word from $\{r_1, \ldots, r_{n\text{-}1}\}^*$, where the index of the symbols r_i are obtained form reading the r-block diagram from the topmost row to the bottom most row, going on each row from left-to-right. Similarly,

we define the column reading of an r-block diagram to the word from $\{r_1, \ldots, r_{n\text{-}1}\}^*$, where the index of the symbols r_i are obtained form reading the r-block diagram from the leftmost column to the rightmost column, going on each column from top-to-bottom.
The words in case (1) of the definition of the set N_{PR} correspond to the row reading of the r-block diagram of the respective pair of sequences satisfying the r-block condition.
Note that the words in case (2) of the definition of the set N_{PR} corresponds to the row reading, now going from the bottom most row to the topmost row, and on each row reading from right-to-left, of the l-block diagram of the respective pair of sequences satisfying the l-block condition.
In the example above the r-block diagram has row reading $r_8r_9\, r_4r_5r_6r_7r_8\, r_3r_4r_5r_6\, r_1r_2r_3r_4r_5$ and column reading $r_1\, r_4r_3r_2\, r_5r_4r_3\, r_6r_5r_4\, r_8r_7r_6r_5\, r_9r_8$.

Lemma 6.2.20. *For any r-block diagram the row reading and the column reading represent the same element of $\mathcal{PR}_n$.*

Proof. It is clear form the graphical representation of the elements of $\mathcal{PR}_n$ that any two elements $\overline{r_i}, \overline{r_j}$, with $|i-j| > 1$, commute, that is, $\overline{r_i}\overline{r_j} = \overline{r_j}\overline{r_i}$. So, fixing a box, filled with a symbol i, on the r-block diagram the letter r_i will commute with every other symbol r_j, with j being a symbol in a box positioned to the northeast.
Hence, consider the symbol, say i_0, at the top of the leftmost column. Then r_{i_0} will commute with every other symbol r_j, where j is in a box to the northeast. So, if we take the row reading of the r-block diagram, which will include the letter r_{i_0}, it will be equal in $\mathcal{PR}_n$, to the word where the letter r_{i_0} is moved to the beginning of the word.
Now, consider new the r-block diagram where we remove the box (with content i_0) we have now considered. Proceed in the same way as in the previous paragraph. Using this procedure, the word that we obtain with the indexes that are being removed from the r-block diagram, is precisely the column reading of the r-block diagram. Thus the two words represent the same element of $\mathcal{PR}_n$. □

Note that the column reading of an r-block diagram corresponds to the reading that is smaller with respect to the lexicographic order. Also the row reading of an l-block diagram is smaller with respect to the lexicographic order.
Let N_{PR} be the subset of A^*, where the words have the form as in the set N, but where the condition (1) is replaced by the following condition:

1′ the column reading of a pair of sequences satisfying the r-block condition.

The following result follows from the graphical representation of elements of $\mathcal{PR}_n$ and the previous lemma.

Proposition 6.2.21. *The set N_{PR} is in one-to-one correspondence with the elements of $\mathcal{PR}_n$, under the mapping ϕ.*

6.3 A complete presentation for $\mathcal{PR}_n$

To review the definition and some properties of rewriting systems, refer to Section 2.7. Consider the following relations on the alphabet A: for any $1 \leq i \leq j < n$,

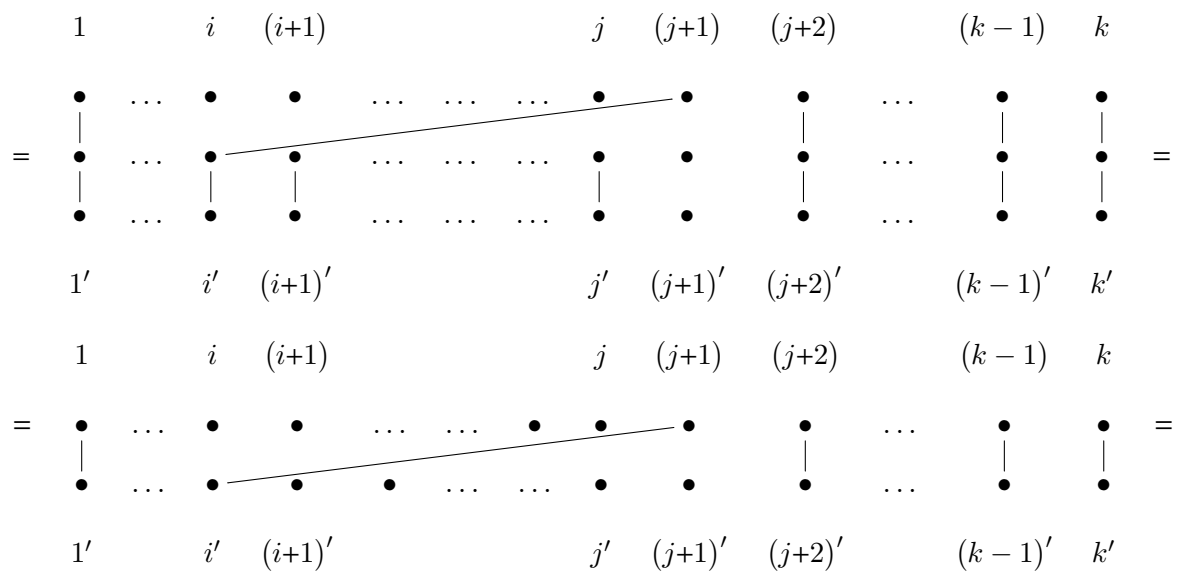
(PR*Comm*) $x_j y_i \to y_i x_j$, for any $x, y \in \{r, l, e\}$ and $|i - j| > 1$;

(PR1)-(PR4) $i < j$	**(PR5)-(PR9)** $i < j$
$r_i\, r_j \dots r_i \to r_i\, r_j \dots r_{i+1}$	$r_j \dots r_i\, r_j \to r_{j\text{-}1} \dots r_i\, r_j$
$l_i\, r_j \dots r_i \to e_i\, r_j \dots r_{i+2}$ $(j > i{+}1)$	$r_j \dots r_i\, l_j \to r_{j\text{-}2} \dots r_i\, e_{j+1}$ $(j > i{+}1)$
$e_i\, r_j \dots r_i \to e_i\, r_j \dots r_{i+1}$	$r_j \dots r_i\, e_{j+1} \to r_{j\text{-}1} \dots r_i\, e_{j+1}$
$x_i\, l_j \dots l_i \to l_j \dots l_i$ $(x \in \{r, l, e\})$	$l_j \dots l_i\, x_j \to l_j \dots l_i$ $(x \in \{r, l\})$
	$l_j \dots l_i\, e_{j+1} \to l_j \dots l_i$

(PR10)-(PR20) $i < j$ and $\lvert i - j\rvert = 1$	**(PR21)-(PR29)** $\lvert i - j\rvert = 0$
$r_i l_{i+1} \to e_i l_{i+1}$	$r_i r_i \to e_i e_{i+1}$
$l_i r_{i+1} \to l_i e_{i+2}$	$l_i l_i \to e_i e_{i+1}$
$r_i e_{i+1} \to e_i e_{i+1}$	$e_i e_i \to e_i$
$l_i e_{i+1} \to l_i$	
	$r_i l_i \to e_{i+1}$
$r_{i+1} l_i \to e_i r_{i+1}$	$l_i r_i \to e_i$
$r_{i+1} e_i \to e_i r_{i+1}$	
$l_{i+1} r_i \to r_i e_{i+2}$	$e_i r_i \to e_i e_{i+1}$
$l_{i+1} e_i \to e_i l_{i+1}$	$e_i l_i \to l_i$
$e_{i+1} r_i \to r_i$	$r_i e_i \to r_i$
$e_{i+1} l_i \to e_i e_{i+1}$	$l_i e_i \to e_i e_{i+1}$
$e_{i+1} e_i \to e_i e_{i+1}$	

Let us denote by R the set of all relations **(PR*Comm*)** and **(PR1)-(PR29)**. Now, we need to prove that all these relations hold in $\mathcal{PR}_n$. The verification process is straightforward but long. We illustrate that the relations (**PR9**) and (**PR14**) in the following paragraphs. We leave it up to the reader to carry out the remaining relations checks.
For example, the diagram for $l_{ij} e_{j+1} \to l_{ij}$ (**PR9**) looks like,

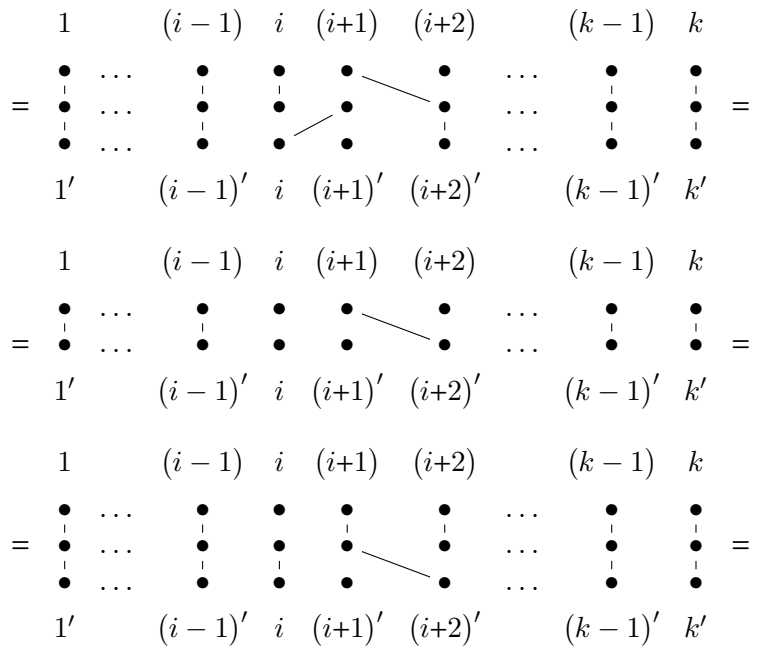
$\overline{l_{ij}}\,\overline{e_{j+1}}$ =



$= \overline{l_{ij}}$

and the diagram for $r_{i+1}l_i \to e_i r_{i+1}$ (**PR14**) has the following appearance:

$\overline{r_{i+1}l_i} =$



$= \overline{e_i r_{i+1}}$.

From the previous considerations, the following lemma holds:

Lemma 6.3.1. *$R \subseteq \ker \phi$.*

Lemma 6.3.2. *R is noetherian.*

Proof. Let us denote by $>$ the ShortLex order on the set A^*, where the order $>$ on the alphabet A is given by

$$r_{n-1} > l_{n-1} > e_n > r_{n-2} > l_{n-2} > e_{n-1} > \cdots > r_2 > l_2 > e_3 > r_1 > l_1 > e_2 > e_1.$$

It is well known that the ShortLex order is a total order on A^* and is well-founded (i.e., there are no infinite descending sequences). Recall that in ShortLex the words are sorted by the length of the words and, in case the two words have the same length, we use the order $>$ on A given above, going from left-to-right on the words.
It is straightforward to check that for any relation $u, v \in R$, we have $u > v$. Since ShortLex is compatible with left and right concatenation, we deduce that whenever $u \to v$ we have $u > v$. Thus R is noetherian as required. □

To make the application of the rewriting rules more comprehensible for the reader, let us provide an example demonstrating the use of the rules mentioned.

Example 6.3.3. *Consider the word w given by $l_8l_9r_4r_5r_6r_7r_8r_3r_4l_1l_2l_3l_4l_5$.*
*Our objective is to rewrite this word in normal form by applying the relations **(PRComm)** and **(PR1)-(PR29)**.*
Let's apply the rewriting rules sequentially, from left to right in the word.
Recall that any element of $\mathcal{PR}_n$ can be uniquely written as the product of maximal r-blocks, maximal l-blocks, and e_i's, in such a way that the indexes are in ascending order.
*We apply the relations to the word w from left to right. The element l_9 can move along in the word until we find the element r_8, successively applying the rule **(PRComm)**.*

$$l_8\mathbf{l_9}r_4r_5r_6r_7r_8r_3r_4l_1l_2l_3l_4l_5 \xrightarrow{*} l_8r_4r_5r_6r_7l_9r_8r_3r_4l_1l_2l_3l_4l_5.$$

*Applying the rule $l_{i+1}r_i \to r_ie_{i+2}$ **(PR16)** in l_9r_8, we have the following reduction:*

$$l_8r_4r_5r_6r_7\mathbf{l_9r_8}r_3r_4l_1l_2l_3l_4l_5 \to l_8r_4r_5r_6r_7\mathbf{r_8e_{10}}r_3r_4l_1l_2l_3l_4l_5.$$

Now, we have used some relations to make the successive reductions and the elements that have undergone the reductions are identified in bold, as follows:

$$\mathbf{l_8}r_4r_5r_6r_7r_8e_{10}r_3r_4l_1l_2l_3l_4l_5 \xrightarrow{*} r_4r_5r_6\mathbf{l_8r_7}r_8e_{10}r_3r_4l_1l_2l_3l_4l_5 \to r_4r_5r_6r_7\mathbf{e_9r_8}e_{10}r_3r_4l_1l_2l_3l_4l_5 \to$$

$$r_4r_5r_6r_7r_8e_{10}\mathbf{r_3r_4}l_1l_2l_3l_4l_5 \xrightarrow{*} r_4r_3r_5r_4r_6r_7r_8e_{10}\mathbf{l_1l_2l_3l_4l_5} \xrightarrow{*} l_1r_4r_3l_2r_5r_4l_3l_4r_6l_5r_7r_8e_{10}.$$

*The rewriting rules used, identified in the sequence below are: **(PRComm)**, **(PR16)** and **(PR18)**.*
Thus, the word $l_1r_4r_3l_2r_5r_4l_3l_4r_6l_5r_7r_8e_{10}$ is equivalent to the initial one; it is merely expressed in normal form according to the rewriting system R.
To verify the accuracy of the applied rules, we can represent both words on a diagram and ensure their equivalence.

Lemma 6.3.4. $N_{PR} = Irr\,(R)$.

Proof. We begin by showing that $N_{PR} \subseteq Irr\,(R)$. Consider a word of the form

$$w_{\left[s_1,s_1^*\right]} \cdots w_{\left[s_t,s_t^*\right]},$$

as in the description of N_{PR}. Notice that $s_i^* < s_{i+1}$. If $w_{\left[s_i,s_i^*\right]}$ is a word corresponding to the column reading of a pair of sequences satisfying the r-block condition, then the letter

$r_{s_i^*}$ is not in the word $w_{\lfloor s_i, s_i^* \rceil}$. The same occurs if $w_{\lfloor s_i, s_i^* \rceil}$ is a word corresponding to a pair of sequences satisfying the l-block condition. Furthermore, if $w_{\lfloor s_l, s_l^* \rceil}$ is of types 1 or 2, in the definition of N_{PR}, and $w_{\lfloor s_{l+1}, s_{l+1}^* \rceil}$ is of type 3, then $s_i^*+1 < s_{i+1}$. Hence, the left hand-side of a relation from R can not be applied in such a way that it overlaps the two words $w_{\lfloor s_i, s_i^* \rceil}$ and $w_{\lfloor s_{i+1}, s_{i+1}^* \rceil}$.
Now, suppose $w_{\lfloor s_i, s_i^* \rceil}$ is a word corresponding to the column reading of a pair of sequences satisfying the r-block condition. A reading of a column is irreducible as $r_j \dots r_i$, with $i < j$, is not the left-hand side of a relation in R. Attending to the description of the r-block condition, the left-hand side of the relations **(PR1)-(PR9)**, can not be applied in such a way that it overlaps two readings of columns. Also note that the left-hand side of the relations **(PR21)**, can not be applied. A similar reasoning can be applied to words $w_{\lfloor s_i, s_i^* \rceil}$ corresponding to a pair of sequences satisfying the l-block condition. We have shown that $N_{PR} \subseteq Irr\,(R)$.
To prove the converse inclusion we will do some observations. We shall refer to r, l and e as the types of generators. An irreducible element satisfies the following properties:

a) the indexes of two consecutive generators can not be equal;

b) it is not possible to have a factor $x_j y_i$, for $x, y \in \{r, l, e\}$, with $j > i$+1;

c) the indexes of two consecutive generators from a different type can not be an i+1 followed by an i;

d) it is not possible to have a factor $e_{i+1} e_i$;

e) it is possible to have a factor $x_j \dots x_i$, with $j > i$, but there is no factor of the form $y_i x_j \dots x_i$ nor of the form $x_j \dots x_i y_j$, for any $y \in \{r, l, e\}$;

f) there is no factor of the form $x_j \dots x_i e_{j+1}$.

Indeed, part a) follows from relations of the form **(PR21-PR29)**; part b) follows from relations of the form **(PR*Comm*)**; part c) follows from relations of the form **(PR14-PR19)**; part d) follows from relations of the form **(PR20)**; part e) follows from relations of the form **(PR1-PR9)**; and part f) follows from relations of the form **(PR7)** and **(PR9)**.
The above properties show that an irreducible word must have the form

$$w_{\lfloor s_1, s_1^* \rceil} \cdots w_{\lfloor s_t, s_t^* \rceil},$$

with $s_1 \leq s_1^* < s_2 \leq s_2^* < \cdots < s_t \leq s_t^*$ and $s_l, s_l^* \in \mathbf{n}$, where each $w_{\lfloor s_l, s_l^* \rceil}$ can be either an element of $\{r_1, \dots, r_{n\text{-}1}\}^*$, $\{l_1, \dots, l_{n\text{-}1}\}^*$ or some e_i with $s_l = s_l^* = i$.
Now notice that if $w_{\lfloor s_l, s_l^* \rceil}$ is an element of $\{r_1, \dots, r_{n\text{-}1}\}^*$, then the subscripts must be in increasing order except when we have a column, that is, factor $x_j \dots x_i$, with $j > i$, of consecutive generators. If we have two consecutive columns, say $x_p \dots x_q\, x_j \dots x_i$, with $j > i$ and $p > q$, then $q < i$ from relations **(PR1)** and **(PR*Comm*)**; and from relations **(PR1)** and **(PR*Comm*)** we deduce that $p < j$. Thus two consecutive columns will either be in

distinct maximal r-blocks, when $q < p$, or they will be part of the same r-block. So each $w_{\left[s_l, s_l^*\right]}$ will correspond to the column reading of a pair of sequences satisfying the r-block condition.
A similar reasoning shows that if $w_{\left[s_l, s_l^*\right]}$ is an element of $\{l_1, \ldots, l_{n\text{-}1}\}^*$, then is a sequence $l_{i_1 j_1} \ldots l_{i_k j_k}$, with $\left((i_1, \ldots, i_k), (j_1, \ldots, j_k)\right)$ satisfying the l-block condition.
This shows that $Irr\,(R) \subseteq N_{PR}$. □

Lemma 6.3.5. *The restriction of ϕ to $Irr\,(R)$ is injective.*

Proof. From the previous lemma we know that $N_{PR} = Irr\,(R)$. Therefore, the result follows from Proposition 6.2.21, since ϕ has a one-to-one correspondence with $\mathcal{PR}_n$, under the mapping ϕ. □

From Proposition 2.7.2, we conclude that R is a complete rewriting system that defines $\mathcal{PR}_n$.

Theorem 6.3.6. *The rewriting system R on the generating set A defines a complete presentation of the monoid $\mathcal{PR}_n$.*

6.4 Another presentation of $\mathcal{PR}_n$

A known presentation for the planar rook monoid also denoted $\mathcal{POI}_n$ in [9], was studied by Fernandes and is found in [9], is as follows:

Theorem 6.4.1. *The planar rook monoid (also denoted $\mathcal{POI}_n$) is defined by the presentation $\langle X \mid R \rangle$, where*

$$X = \{x_0 := \left[n-1, 1\right]^r,\ x_1 := l_2,\ \ldots\ , x_{n-1} := l_n\ \}$$

and R given by the rules:

(R1)	$x_i x_0 = x_0 x_{i+1}$,	$1 \leq i \leq n-2$;
(R2)	$x_j x_i = x_i x_j$,	$2 \leq i{+}1 \leq n-1$;
(R3)	$x_0^2 x_1 = x_0^2 = x_{n-1} x_0^2$,	$2 \leq i{+}1 \leq n-1$;
(R4)	$x_{i+1} x_i x_{i+1} = x_{i+1} x_i = x_i x_{i+1} x_i$,	$1 \leq i \leq n-2$;
(R5)	$x_i x_{i+1} \ldots x_{n-1} x_0 x_1 \ldots x_{i-1} x_i = x_i$,	$0 \leq i \leq n-1$;
(R6)	$x_{i+1} \ldots x_{n-1} x_0 x_1 \ldots x_{i-1} x_i^2 = x_i^2$,	$1 \leq i \leq n-1$;

where $x_i \ldots x_j = \varepsilon$ if $j < i$.
Furthermore, $\mathcal{POI}_n$ has rank n.

However, in contrast to the complete presentation for $\mathcal{PR}_n$ described above, this one is not complete.

Proposition 6.4.2. *The rewriting system R is not complete.*

Proof. For a rewriting system to define a complete presentation, it is necessary for the rewriting system to satisfy two properties: being confluent and noetherian.

Orienting the relations as given in the previously from left to right, it is easy to verify that R is noetherian.
However, it is not confluent. The confluence property suggests that by reducing the same word in different ways, both reductions have to converge on the same word. In this case, that is not the situation; applying the necessary rules to each of the reductions will result in different words.
For example, the rule used to arrive at the contraction is **(R4)**, $x_{i+1}x_ix_{i+1} = x_{i+1}x_i$. Orienting the relations as given previously from left to right, we have $x_{i+1}x_ix_{i+1} \to x_{i+1}x_i$.
Thus,

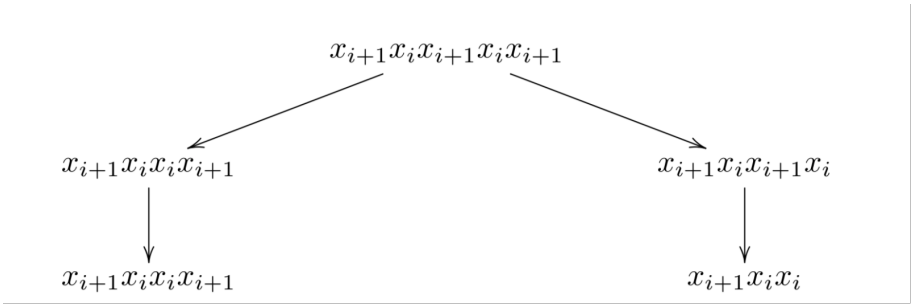


Figure 6.7: Scheme for checking the confluence property for the **(R4)** rule.

Conclude that, as the system did not verify the confluence property, the rewriting system R is not complete. □

7

Conclusions

In this thesis we study cryptographic key-exchange protocols that use algebraic structures for key encoding. In particular, we were interested in the cryptography scheme for the Stickel's key exchange protocol. The original protocol was introduced by Stickel ([34]). The main objective of our work is to propose a new platform for the protocol under study.

What was done

At the beginning of the fourth chapter, we recalled Stickel's protocol introduced by Stickel ([34]). We wrote Python code to implement Stickel's key exchange protocol and construct the linear algebra attack by Myasnikov et al. ([29]).
We ran the code to test and benchmark its execution time, which revealed a possible exponential relationship with input size. In our tests, only a non-trivial solution was found, allowing us to experimentally corroborate the results of Myasnikov.
Since Stickel's key exchange protocol maintains weak security, we decided to investigate the possibility of using the partition monoid as a platform for this protocol. As emphasised throughout the thesis, obtaining normal forms is important because they are necessary for considering the partition monoid as a platform for Stickel's key exchange protocol.
In the fifth chapter, recognising the significance of normal forms, we began by studying a presentation given by James East for the partition monoid ([8]). In this presentation, East proposes the existence of natural factorisation for the partition monoid. The factorisation comprises three submonoids: $\mathcal{L}_n$, the submonoid of the full transformation semigroup; $\mathcal{I}_n$, the symmetric inverse semigroup; and $\mathcal{R}_n$, an anti-isomorphic copy of $\mathcal{L}_n$.
We concluded that the monoid $\mathcal{I}_n$ is not confluent, and hence not complete.
Given the result obtained, it was not possible for us to derive the normal forms from East's presentation. Therefore, we decided to study a submonoid of the partition monoid, the planar rook monoid, with the aim of providing a complete presentation for it. The importance of having a complete presentation lies in the existence of normal forms.
In the last chapter we studied the planar rook monoid and successfully constructed a complete presentation for it. Additionally, we evaluated another presentation proposed by Fernandes ([9]), concluding that it is not compete. This achievement not only addresses the

limitations encountered with East's presentation but also establishes a robust foundation for future research endeavours in cryptographic protocol design and analysis.
In conclusion, our thesis makes significant strides in advancing both theoretical understanding and practical application in the fields of semigroup theory and cryptography. By identifying vulnerabilities in existing protocols and exploring novel approaches to address them, we contribute to the ongoing effort of developing secure cryptographic systems.

Inspiration for further research

In conclusion, our thesis has established a foundation for future research in cryptographic key exchange protocols. Future efforts can be concentrated on key objectives.
Firstly, to discover complete presentations for the submonoids $\mathcal{L}_n$ (and $\mathcal{R}_n$) and $\mathcal{I}_n$.
Subsequently, constructing a complete rewriting system for the partition monoid using the normal forms given from the complete rewriting system of the partition monoid, aiming to implement Stickel's protocol using this new platform.
Lastly, conducting an evaluation of the security of the proposed protocol, ensuring its resilience and efficacy in real-world cryptographic scenarios.
By pursuing these future research directions, it will contribute to the advancement of cryptographic theory and the development of secure communication protocols in digital environments.

Bibliography

[1] I. J. Aalbersberg and H. J. Hoogeboom. "Characterizations of the decidability of some problems for regular trace languages". In: *Math. Systems Theory* 22.1 (1989), pp. 1–19. ISSN: 0025-5661. DOI: 10.1007/BF02088289. URL: https://doi.org/10.1007/BF02088289 (cit. on p. 9).

[2] I. Anshel, M. Anshel, and D. Goldfeld. "An algebraic method for public-key cryptography". In: *Mathematical Research Letters* 6 (1999), pp. 287–291. URL: https://api.semanticscholar.org/CorpusID:11621019 (cit. on p. 15).

[3] R. Brauer. "On algebras which are connected with the semisimple continuous groups". In: *Ann. of Math. (2)* 38.4 (1937), pp. 857–872. ISSN: 0003-486X. DOI: 10.2307/1968843. URL: https://doi.org/10.2307/1968843 (cit. on p. 29).

[4] A. J. Cain. "Nine Chapters on the Semigroup Art". In: *Lecture notes for M 431 Semigroups* (2013) (cit. on pp. 6, 7).

[5] A. Carvalho. "On generalized conjugacy and some related problems". In: *Comm. Algebra* 51.8 (2023), pp. 3528–3542. ISSN: 0092-7872. DOI: 10.1080/00927872.2023.2186132. URL: https://doi.org/10.1080/00927872.2023.2186132 (cit. on p. 9).

[6] W. Diffie and M. E. Hellman. "New directions in cryptography". In: *IEEE Trans. Inform. Theory* IT-22.6 (1976), pp. 644–654. ISSN: 0018-9448. DOI: 10.1109/tit.1976.1055638. URL: https://doi.org/10.1109/tit.1976.1055638 (cit. on p. 17).

[7] J. East. "A presentation for the singular part of the full transformation semigroup". In: *Semigroup Forum* 81.2 (2010), pp. 357–379. ISSN: 0037-1912. DOI: 10.1007/s00233-010-9250-1. URL: https://doi.org/10.1007/s00233-010-9250-1 (cit. on pp. 33, 34).

[8] J. East. "Generators and relations for partition monoids and algebras". In: *J. Algebra* 339 (2011), pp. 1–26. ISSN: 0021-8693. DOI: 10.1016/j.jalgebra.2011.04.008. URL: https://doi.org/10.1016/j.jalgebra.2011.04.008 (cit. on pp. 2, 32, 38, 55).

[9] V. H. Fernandes. "The monoid of all injective order preserving partial transformations on a finite chain". In: *Semigroup Forum* 62.2 (2001), pp. 178–204. ISSN: 0037-1912. DOI: 10.1007/s002330010056. URL: https://doi.org/10.1007/s002330010056 (cit. on pp. 41, 53, 55).

[10] D. G. Fitzgerald and J. Leech. "Dual symmetric inverse monoids and representation theory". In: *Journal of the Australian Mathematical Society. Series A. Pure Mathematics and Statistics* 64.3 (1998), pp. 345–367. DOI: 10.1017/S1446788700039227 (cit. on p. 29).

[11] F. Goodman and H. Wenzel. "The Temperly Lieb algebra at roots of unity". In: *Pacific J. Math.* 161 (1993), pp. 307–334 (cit. on p. 29).

[12] J. J. Graham and G. I. Lehrer. "Cellular algebras". In: *Invent. Math.* 123.1 (1996), pp. 1–34. ISSN: 0020-9910. DOI: 10.1007/BF01232365. URL: https://doi.org/10.1007/BF01232365 (cit. on p. 29).

[13] T. Halverson and A. Ram. "Partition algebras". In: *European J. Combin.* 26.6 (2005), pp. 869–921. ISSN: 0195-6698. DOI: 10.1016/j.ejc.2004.06.005. URL: https://doi.org/10.1016/j.ejc.2004.06.005 (cit. on p. 29).

[14] M. Hellman. "An overview of public key cryptography". In: *IEEE Communications Magazine* 40.5 (2002), pp. 42–49. DOI: 10.1109/MCOM.2002.1006971 (cit. on p. 17).

[15] P. M. Higgins. "Techniques of semigroup theory". In: *The Clarendon Press, Oxford University Press, New York* (1992) (cit. on p. 29).

[16] J. M. Howie. "An introduction to semigroup theory". In: *Academic Press [Harcourt Brace Jovanovich, Publishers], London-New York* (1976) (cit. on p. 29).

[17] V. F. R. Jones. "A quotient of the affine Hecke algebra in the Brauer algebra". In: *Enseign. Math. (2)* 40.3-4 (1994), pp. 313–344. ISSN: 0013-8584 (cit. on p. 29).

[18] K. H. Ko et al. "New Public-Key Cryptosystem Using Braid Groups". In: *Advances in Cryptology — CRYPTO 2000*. Ed. by M. Bellare. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 166–183 (cit. on p. 15).

[19] M. Ladra and P. V. Silva. "The generalized conjugacy problem for virtually free groups". In: *Forum Math.* 23.3 (2011), pp. 447–482. ISSN: 0933-7741. DOI: 10.1515/FORM.2011.015. URL: https://doi.org/10.1515/FORM.2011.015 (cit. on p. 9).

[20] S. Lipscomb. "Symmetric Inverse Semigroups". In: 1996. URL: https://api.semanticscholar.org/CorpusID:118680607 (cit. on p. 29).

[21] M. Lohrey. "The rational subset membership problem for groups: a survey". In: *Groups St Andrews 2013*. Vol. 422. London Math. Soc. Lecture Note Ser. Cambridge Univ. Press, Cambridge, 2015, pp. 368–389 (cit. on p. 9).

[22] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[23] A. Malheiro. "Complete rewriting systems for codified submonoids." In: *Int. J. Algebra Comput.* 15.2 (2005), 207–216. ISSN: 0218-1967; 1793-6500/e. DOI: 10.1142/S0218196705002220. URL: http://dx.doi.org/10.1142/S0218196705002220 (cit. on p. 9).

[24] P. Martin. "Temperley-Lieb algebras for non-planar statistical mechanics - The Partition Algebra construction". In: *Journal of Knot Theory and Its Ramifications* 03 (1994), pp. 51–82. URL: https://api.semanticscholar.org/CorpusID:123668349 (cit. on p. 29).

[25] A. Menezes, P. van Oorschot, and S. Vanstone. "Handbook of applied cryptography". In: *CRC Press Series on Discrete Mathematics and its Applications* (1997) (cit. on pp. 12, 15).

[26] A. Miasnikov and P. Schupp. "Computational complexity and the conjugacy problem". In: *Computability* 6.4 (2017), pp. 307–318. ISSN: 2211-3568. DOI: 10.3233/com-160060. URL: https://doi.org/10.3233/com-160060 (cit. on pp. 15, 16).

[27] C. F. Miller. "On Group-Theoretic Decision Problems and Their Classification. (AM-68)". In: *Princeton University Press* (1971). URL: http://www.jstor.org/stable/j.ctt1b7x83g (cit. on p. 9).

[28] E. H. Moore. "Concerning the Abstract Groups of Order $k!$ and $\frac{1}{2}k!$ Holohedrically Isomorphic with the Symmetric and the Alternating Substitution-Groups on k Letters". In: *Proc. Lond. Math. Soc.* 28 (1896/97), pp. 357–366. ISSN: 0024-6115. DOI: 10.1112/plms/s1-28.1.357. URL: https://doi.org/10.1112/plms/s1-28.1.357 (cit. on pp. 32, 33).

[29] A. Myasnikov, V. Shpilrain, and A. Ushakov. "Group-based Cryptography". In: *Advanced Courses in Mathematics* (2000) (cit. on pp. 1, 10, 13, 16, 20, 28, 55).

[30] L. Popova. "Defining relations in some semigroups of partial transformations of a finite set". In: *Uchenye Zap. Leningrad Gos. Ped. Inst.* 218 (1961), pp. 191–212 (cit. on pp. 36, 37).

[31] V. Shpilrain. "Assessing security of some group based cryptosystems". In: *Group theory, statistics, and cryptography*. Vol. 360. Contemp. Math. Amer. Math. Soc., Providence, RI, 2004, pp. 167–177. DOI: 10.1090/conm/360/06577. URL: https://doi.org/10.1090/conm/360/06577 (cit. on p. 16).

[32] V. Shpilrian. "Problems in group theory motivated by cryptography". preprint on webpage at https://arxiv.org/pdf/1802.07300.pdf. 2018 (cit. on p. 13).

[33] P. V. Silva. "On the rational subsets of the monogenic free inverse monoid". In: *J. Algebra* 618 (2023), pp. 214–240. ISSN: 0021-8693. DOI: 10.1016/j.jalgebra.2022.12.006. URL: https://doi.org/10.1016/j.jalgebra.2022.12.006 (cit. on p. 9).

[34] E. Stickel. "A New Method for Exchanging Secret Keys". In: *Third International Conference on Information Technology and Applications (ICITA'05)*. Vol. 2. 2005, pp. 426–430. DOI: 10.1109/ICITA.2005.33 (cit. on pp. 18–20, 28, 55).

[35] S. Wilcox. "Cellularity of diagram algebras as twisted semigroup algebras". In: *J. Algebra* 309.1 (2007), pp. 10–31. ISSN: 0021-8693. DOI: 10.1016/j.jalgebra.2006.10.016. URL: https://doi.org/10.1016/j.jalgebra.2006.10.016 (cit. on p. 29).

A

Appendices

The code of the Linear Algebra attack on Stickel's protocol in Python:

```
import galois
import random
import numpy as np
from sympy import symbols, Matrix, equation, solve, zeros, and add
from itertools import product

# This function should to addition in F2 (not only numbers but also variables)
def f2add(a, b):
    if a == b:
        return 0
    elif isinstance(a,int) and isinstance(b,int):
        return 1
    else:
        result = a + b
        if isinstance(result, Add):
            for _, coef in result.as_coefficients_dict().items():
                return sum(
                    varia for varia, coef in result.as_coefficients_dict().items() if coef % 2
                )
        # If the result is a symbolic expression with odd coefficients, it returns the sum of the terms odd the coeficients

        return result

# This function should to the product in F2 (not only numbers but also variables)
def f2multiply(a, b):
    if isinstance(a, int) and isinstance(b, int):
        return a * b  # Normal multiplication for integers
    elif isinstance(a, int):
        return b if a == 1 else 0  # If a is 1, return b; otherwise, return 0
    elif isinstance(b, int):
```

```
        return a if b == 1 else 0  # If b is 1, return a; otherwise, return 0

    # This suggests that multiplication by 1 acts as an identity, while multiplication by any other integer results in 0.

    else:
        return a * b  # And operation for symbols


# The function should sum matrices in F2 (not only numbers but also variables)
def matrixaddF2(A, B,dimension):
    result=zeros(dimension,dimension)
    for i in range(dimension):
        for j in range(dimension):
            result[i,j]=f2add(A[i,j],B[i,j])
    return Matrix(result)

# The function should do the product of matrices in F2 (not only numbers bur also variables)
def matrixmultiplyF2(A, B):
    result = []
    for i in range(A.shape[0]):
        row = []
        for j in range(B.shape[1]):
            entry = 0
            for k in range(A.shape[1]):
                entry = f2add(entry, f2multiply(A[i, k], B[k, j]))
            row.append(entry)
        result.append(row)
    return Matrix(result)

# Define the field
GF = galois.GF(2)

# k is the dimension of the matrices
k = 31

n = random.randint(1,2**k-2)
m = random.randint(1,2**k-2)

# Restriction that ab != ba
# Randomly generates an irreducible polynomial of degree k in F2

poly_a = galois.irreducible_poly(2, k, method="random")
while True:
    poly_b = galois.irreducible_poly(2, k, method="random")
    if poly_a != poly_b:
        break
```

```
77
78 # Extract the coefficients to fill in the last column
79 alastcol=galois.Poly.coefficients(polya)
80 blastcol=galois.Poly.coefficients(polyb)
81
82 # Define matrix 'a' as the Stickel matrices
83 a = galois.FieldArray.Zeros((k,k))
84 for i in range(1,k):
85         a[i,i-1] = GF(1)
86
87 for i in range(k):
88     a[i][k-1] = alastcol[-i-1]
89
90 # Define the matrix 'a' in a matrix of F2
91 a=GF(a)
92
93 # Define matrix 'b' as the Stickel matrices
94 b = galois.FieldArray.Zeros((k,k))
95 for i in range(1,k):
96     b[i,i-1] = GF(1)
97
98 for i in range(k):
99     b[i][k-1] = blastcol[-i-1]
100
101 # Define the matrix 'b' in a matrix of F2
102 b=GF(b)
103
104 # Calculate the powers, the matrix 'u' and the inverse of 'u'. The result
      is modulo 2, because we used Galois before
105 a_n = np.linalg.matrix_power(a,n)
106 b_m = np.linalg.matrix_power(b,m)
107 u = np.matmul(a_n,b_m)
108 u_inv = np.linalg.inv(u)
109
110 # Create copies of a^(n), b^(m) and u^(-1) but as vectores of integers.
111 u = np.array(u)
112 a_z = np.array(a)
113 b_z = np.array(b)
114 u_inv_z = np.array(u_inv)
115
116 # Create the symbol matrix
117 Y=Matrix(symbols('Y1:%d' % (k*k + 1))).reshape(k, k)
118
119 # Define the two equations
120 # Since you're multiplying matrices by a matrix Y of symbols/variables, the
      matmul modules don't work.
121 # So we use the function we defined like this to do the multiplication in
     F2 for matrices when the product has Y.
122 # For the rest, we use matmul because it should be more efficient.
```

```
eq1 = Eq(matrixmultiplyF2(Y,np.matmul(u_inv,a)),matrixmultiplyF2(a_z,
    matrixmultiplyF2(Y,u_inv_z)))
eq2 = Eq(matrixmultiplyF2(Y,b_z),matrixmultiplyF2(b_z,Y))

# The sum of both sides of the equation: lhs= left-hand side e rhs= right-
    hand side.
eq1norm=matrixaddF2(eq1.lhs,eq1.rhs,k)
eq2norm=matrixaddF2(eq2.lhs,eq2.rhs,k)

# Create a vector with the variables (not matrix, but vector)
Y_vec=np.array(symbols('Y1:%d' % (k*k + 1)))

# Define the matrix that defines the system Ax=0 (2k^2 x 2k^2)
Msyst=[]
for i in range(k):
    for j in range(k):
        Msyst.append(list((eq1norm.row(i).jacobian(Y_vec)).row(j)))
        Msyst.append(list((eq2norm.row(i).jacobian(Y_vec)).row(j)))

# Msystint receives the matrix Msyst and converts the elements into
    integers and then into elements of F2 and calculate the nullspace.
Msystint = GF([[GF(int(element)) for element in row] for row in Msyst])
nulspac=Msystint.null_space()


if len(nulspac)==1:
    solution=nulspac[0]
    dic_sub= {'Y'+str(i): solution[i-1] for i in range(1,k**2+1)}
    Ysolution = Y.subs(dic_sub)
# This means that there is only one basis vector, or in other words, one
    solution to the equation

else:
    lista= list(product(range(2), repeat=len(nulspac)))
    j=0
    solution=nulspac[j]
    dic_sub= {'Y'+str(i): solution[i-1] for i in range(1,k**2+1)}
    Ysolution = Y.subs(dic_sub)
    while Matrix.det(Ysolution)%2 != 1:
        j+=1
        summands=[]
        for i in range(len(lista[j])):
            if lista[j][i]==1:
                summands.append(i)
        solution=nulspac[0]
        for j in len(summands):
            solution=solution+ nulspac[j]
        dic_sub= {'Y'+str(i): solution[i-1] for i in range(1,k**2+1)}
        Ysolution = Y.subs(dic_sub)
```

```
# This means that there are several basis vectors, indicating several possible solutions.
# It generates all possible combinations of coefficients (0 or 1) for the basis vectors. Then the code iterates through combinations of these basis vectors to try to find a solution matrix whose determinant is odd (in F2 this means that the determinant is 1). This guarantees that the matrix is invertible in F2.

Ysolutionmatrix =np.array(Ysolution)
    return M

# Define the matrix 'Y_solution_matrix' in a matrix of F2
Ysolutionmatrix = GF(Ysolutionmatrix.astype(int))

# x^(-1) = yu^(-1)
x_inv = np.matmul(Ysolutionmatrix,u_inv)

x = np.linalg.inv(x_inv)

# u = xwy =xy
np.matmul(x,Ysolutionmatrix)


# Implement Eve's attack

r = random.randint(1,2**k-2)
s = random.randint(1,2**k-2)

# Calculate the powers
a_r = np.linalg.matrix_power(a,r)
b_s = np.linalg.matrix_power(b,s)

# Calculate v=a^(r)b^(s)
v = np.matmul(a_r,b_s)

v_inv = np.linalg.inv(v)

# The fuction that allows Eve to discover the secret key
def kfind(x,y,q):
    x_q = np.matmul(x,q)
    k_find = np.matmul(x_q,y)
    return k_find

# The function that allows you to build the real secret key
def kshared(a,b,r,s,m,n):
    a_r_n = np.linalg.matrix_power(a,r+n)
    b_s_m = np.linalg.matrix_power(b,s+m)
    k_shared = np.matmul(a_r_n,b_s_m)
    return k_shared
```