CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE COMPUTACIÓN

“Pruebas de conocimiento nulo basados en sistemas de ternas de Steiner”

T E S I S

Que presenta

EDGAR GONZÁLEZ FERNÁNDEZ

para obtener el grado de

DOCTOR EN CIENCIAS EN COMPUTACIÓN

Directores de la Tesis:

Dr. Guillermo Morales-Luna

Dr. Feliú Davino Sagols Troncoso

Ciudad de México FEBRERO, 2020

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

CAMPUS ZACATENCO

DEPARTMENT OF COMPUTER SCIENCE

"Zero-knowledge proofs based on Steiner triple systems"

T H E S I S

submitted by

EDGAR GONZÁLEZ FERNÁNDEZ

for the degree of

PHD IN COMPUTER SCIENCE

Advisors:

Dr. Guillermo Morales-Luna

Dr. Feliú Davino Sagols Troncoso

Mexico City FEBRUARY, 2020

Resumen

El creciente uso de dispositivos electrónicos potentes y la disponibilidad de redes de comunicación que proporcionan conectividad ubicua y de alto rendimiento, permite a diversas aplicaciones digitales transmitir grandes volúmenes de datos en períodos breves de tiempo. En muchas ocasiones, los datos transmitidos requieren de un canal de comunicación protegido mediante procedimientos confiables de autenticación y privacidad basados en esquemas criptográficos. Para su implementación, estos esquemas consideran problemas matemáticos que son difíciles de resolver. Aunque existe una numerosa cantidad de algoritmos criptográficos, sólo unos cuantos son utilizados en el mundo real, como lo son el conocido procedimiento de Rivest-Shamir-Adleman (RSA), y el Estándar de Firma Digital (DSS por sus siglás en inglés), debido principalmente a su resistencia y fácil implementación. Estos algoritmos son la base de varias técnicas de firma digital y protocolos de autenticación e identificación que se utilizan comúnmente en comercio electrónico, transacciones bancarias y servicios gubernamentales, entre otros. Aunado a esto, su uso ha aumentado debido a la introducción de nuevas aplicaciones, como la autenticación multifactorial y las criptodivisas.

El rápido desarrollo de técnicas de criptoanálisis y el auge de la computación cuántica ponen en peligro estas medidas de seguridad, siendo la amenaza más alarmante la existencia de un algoritmo capaz de resolver eficientemente el problema de la factorización, siempre que la construcción de una computadora cuántica sea posible. Estas implicaciones sugieren que deben estudiarse y desarrollarse nuevos mecanismos de seguridad ante la posible materialización de estas amenazas. Recientemente, las pruebas de conocimiento nulo han sido consideradas como una alternativa a los protocolos de autenticación e identificación existentes, siendo la mayor ventaja que estos protocolos se basan en problemas que aún no han sido resueltos por algoritmos cuánticos.

Además de los servicios de autenticación e identificación, tecnologías novedosas como el blockchain y las criptodivisas, que requieren servicios de anonimato, han mostrado las capacidades de las pruebas de conocimiento nulo: una técnica fiable

para demostrar el conocimiento de datos específicos sin revelar detalles; por ejemplo, para demostrar que una cuenta tiene suficiente crédito para comprar un artículo sin conocer el saldo exacto. También se ha manifestado interés en dicha tecnología para la autenticación en el almacenamiento en la nube y en el Internet de las cosas, fomentando el desarrollo de estos protocolos.

En esta tesis nos centramos en la dificultad de encontrar isomorfismos de diseños combinatorios, específicamente sistemas de ternas de Steiner. Se realiza un estudio exhaustivo de las herramientas más importantes para probar isomorfismo. Además, se revelan algunas mejoras de estas técnicas y, con base a las observaciones de dicho análisis, se caracterizan instancias difíciles, las cuales comprenden los elementos básicos para una variedad de protocolos de conocimiento nulo.

Abstract

The increasing use of powerful electronic devices and the availability of communication networks that provide ubiquitous and high performance connectivity, allow digital applications to transmit huge volumes of data in short periods of time. In many cases, the data transmitted requires a communication channel protected by reliable authentication and privacy procedures based on cryptographic schemes. For their implementation, these schemes consider mathematical problems that are difficult to solve. Although several cryptographic algorithms exist, only a few of them are used in real world applications, such as the well-known Rivest-Shamir-Adleman procedure (RSA), and the Digital Signature Standard (DSS), mainly due to their robustness and easy implementation. These algorithms are the basis of several digital signature techniques, and authentication and identification protocols that are commonly used in e-commerce, banking transactions and government services, among others. In addition, their use has increased due to the introduction of new applications, such as multi-factor authentication and cryptocurrencies.

The rapid development of cryptanalysis techniques and quantum computers jeopardizes these security measures, the most alarming threat being the existence of an algorithm capable of efficiently solving the factorization problem, whenever the construction of a quantum computer is possible. These implications suggest that new security measures should be studied and developed in view of the possible materialization of these threats. Recently, zero-knowledge tests have been considered as an alternative to existing authentication and identification protocols, the major advantage being that these protocols are based on problems that have not yet been solved by quantum algorithms.

In addition to authentication and identification services, novel technologies such as blockchain and cryptocurrencies, which require anonymity services, have shown the capabilities of zero-knowledge proofs: a reliable technique to demonstrate knowledge of specific data without revealing details; for example, to demonstrate that an account has enough credit to purchase an item without knowing the exact balance. Interest

has also been expressed in such technology for authentication in cloud storage and on the Internet of Things, encouraging the development of these protocols.

In this thesis we focus on the difficulty of finding isomorphisms of combinatorial designs, specifically of Steiner triple systems. An exhaustive study of the most important tools for testing isomorphism is carried out. In addition, some improvements of these techniques are revealed and, based on the observations of this analysis, difficult instances are characterized, which comprise the basic elements for a variety of zero-knowledge protocols.

Dedicatoria

Para mi esposa e hija, por una etapa que apenas comienza.

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnologa (CONACyT) por el sustento económico ofrecido por el programa de becas nacionales, sin el cual no sería posible lograr este objetivo.

A mis asesores, los Drs. Guillermo Morales y Feliú Sagols, por el apoyo que me han brindado a lo largo de todo este proceso.

Al personal administrativo, en especial Sofía Reza, Erika Ríos y Felipa Rosas, de quienes siempre he recibido apoyo total.

A mis sinodales: los Drs. Guadalupe Rodríguez, Juan Carlos Ku, Oliver Schütze, y Rafael Heraclio Villarreal, por los aportes y apoyo en la recta final de este proceso.

Finalmente a mis padres, que han estado conmigo en cada etapa y logro de mi vida.

Contents

List of Figures **X**

List of Tables **XI**

List of Algorithms **XIII**

1 Introduction **1**
- 1.1 Related work . . . 2
- 1.2 Objectives . . . 3
- 1.3 Research products . . . 4
- 1.4 Outline . . . 5

2 Preliminaries and early work **7**
- 2.1 Graph theory . . . 7
- 2.2 Interactive and zero-knowledge proof systems . . . 8
- 2.3 Quasigroups . . . 9
- 2.4 Steiner triple systems . . . 9
- 2.5 Early work . . . 12
  - 2.5.1 Implementation of cryptographic schemes based on multivariate cryptography . . . 12
  - 2.5.2 A zero-knowledge proof system based on an algebraic interpretation of the graph isomorphism problem . . . 16

3 Steiner triple systems **21**
- 3.1 Subsystems and partial systems . . . 21
  - 3.1.1 Embedding systems and defining sets . . . 22
- 3.2 Cycle Graphs . . . 23
  - 3.2.1 The switching transformation . . . 26
  - 3.2.2 Effect of switching on cycle graphs . . . 29

3.3 Configurations 32
3.4 Trades 33

4 The Isomorphism Problem for STS 35
4.1 Miller's Algorithm 35
4.2 Algorithm based on cycle graphs 37
4.2.1 Complexity of the algorithm 40
4.2.2 Joining cycle graphs 42
4.3 Improving Miller's algorithm 44

5 Building difficult instances 49
5.1 Hall triple systems 50
5.1.1 Hill-climbing algorithm for HTS 51
5.1.2 Algebraic construction of HTS 54
5.2 Non-uniform instances 55
5.3 Experimental results 56

6 Zero Knowledge Proof Systems Based on STSs 61
6.1 A ZKP based on isomorphism problem 62
6.2 A ZKP based on non-isomorphism problem 63
6.3 A ZKP based on the switching transformation 64
6.3.1 Authentication protocol 64
6.3.2 Cryptanalysis 67
6.4 Implementation issues 67

7 Conclusions and future work 69

References 71

List of Figures

2.1 Representations of a STS(7) 10
2.2 A solution to Kirkman's problem 11
2.3 Process to generate the polynomials set associated to the graph isomorphism. 17
2.4 Isomorphism composition and resulting systems. 17

3.1 Cycle graph of the STS(15) example. 24
3.2 Cycles of length 4 obtained from a quadrilateral. 26
3.3 Visual representation of the cycle switch transform. 27
3.4 Effect of sw on cycle graphs. 31
3.5 Joining two cycles with a switching transformation. 32

4.1 Cycle graphs for isomorphism technique. 41
4.2 Solving isomorphism with cycle graphs. 43

6.1 Cycle switching in isomorphic graphs. 64

List of Tables

3.1 A Steiner triple system of size 15. 23
3.2 Small configurations in Steiner triple systems. 33

5.1 Dimension vs order. 51
5.2 Computational resources. 57
5.3 Results of executions of isomorphism tests. 58
5.4 Results of executions of isomorphism tests. 58

List of Algorithms

1 Miller's Algorithm for Quasigroup Isomorphism 36
2 Extending a partial bijection 38
3 Finding isomorphisms with cycle graphs 39
4 Finding isomorphisms with cycle graphs 44
5 Getting a generator set 46
6 Improved version of Miller's algorithm 47
7 Hill-climbing algorithm for construction of Steiner triple systems . . . 52
8 Hill-climbing algorithm for construction of Hall triple systems 53

Chapter 1

Introduction

Considering recent advances in quantum computing and the threat it represents for most of the security schemes employed in communication systems worldwide, it is evident that novel and existing security measures must be proposed and analyzed. In our original approach, we intended to analyze security schemes based on the *Multivariate Quadratic Problem* ($\mathcal{MQ}$), which consists of solving a system of multivariate quadratic polynomial equations on n variables. Some advances in this direction include the development of a software package that allows performing cryptographic operations using existing digital signature schemes based on $\mathcal{MQ}$. A second approach consisted in designing a zero-knowledge proof (ZKP) system based on solving a system of quadratic polynomials built from a translation of the isomorphism problem on graphs. Both works will be addressed briefly in Chapter 2. The analysis performed on the latter approach led to defining a new direction for the research, which helped to delineate the main topic for this work. In the search for difficult instances for the graph isomorphism problem, some problems involving combinatorial designs turned out to be related, in particular, the isomorphism in Steiner triple systems. In addition to the isomorphism problem, further topics such as the problem inverting successive cycle switch transformations (introduced in 3.2.1) in Steiner triple systems have been considered. These approaches, which are not based on the $\mathcal{MQ}$ problem anymore, have been useful in the development of other zero-knowledge protocols, introduced in Chapter 6.

Though the cryptographic application is an important issue and, in fact, is one of the motivations of the present work, it is worth mentioning that the main results are also of interest from a combinatorial point of view: the central problem to be studied is the isomorphism problem on Steiner triple systems (STS). Our main goal is to provide difficult instances of this problem, addressed in detail in Chapter 4. Briefly, a

STS consists of a set X of n elements and a second set, conformed by triples of X in such a way that any possible pair of elements in X is contained in exactly one triple. More details about STSs will be introduced in Chapter 3.

1.1 Related work

Steiner triple systems, and in general, block designs have been employed in several areas of science, from the design of experiments to coding theory. The first registers of known applications of block designs date back to several centuries. One of the first appearances was the construction of *latin squares*, a matrix arrangement of size $n \times n$ where each row and column consists of n different elements. The first registered construction is attributed to Choi Seok-Jeong in 1700 [9, I.2], and his main goal was strongly related to the construction of magic squares. However, there is a more prominent association of block designs with statistical applications, specifically related to the design of experiments [16, 17]. In this context, a block refers to a sub-sample of the universe. Each possible trial can be performed on a block to study the effects according to the experiment carried out.

A closer use of combinatorial designs with the work here developed is related to coding theory, particularly useful to study perfect codes [4], some of them generated from Steiner triple systems [41]. Applications to further communication services can be found in regard to optic fiber channels, where codes based on block designs are proposed to allow synchronous and asynchronous communication between many users over a common wide-band channel [10].

With regards to cryptographic applications, in [54], Stinson and Vanstone create a scheme for secret sharing. Briefly, by considering a collection disjoint systems $S_1, \ldots, S_m$, i.e., with no block in common, a block can be shared among three parties by assigning an element to each one. To decide the source of such a block, the three parties can share the information and look at the S_i containing it. Since any pair is contained precisely in a block, the combined information of any two or fewer parties is not useful to retrieve the secret, in this case, the sub-index of the STS containing the block. This idea, which is formalized in [54], is based on the idea of partitioning a Steiner system $S(t, w, v)$ in subsystems $STS(t-1, w, v)$. In short, a Steiner system $S(t, w, v)$ is a generalization of the definition for STSs, which consists of a set X of size v and a block set $\mathcal{B}$ conformed by subsets of size w, such that any subset of size t is contained in exactly one $B \in \mathcal{B}$. A STS is Steiner system with parameters $t = 2$ and $w = 3$.

In [50], a secret sharing protocol based on minimal defining sets (MDS), was introduced. In short, a MDS is a minimal subset of the set of blocks $\mathcal{B}$, which extends uniquely to $\mathcal{B}$. Again, by considering a collection of m STSs, in this case, with known minimal designs, one sub-index is chosen as the secret. The MDS of the selected STS is split by distributing blocks among the involved parties. Since no subset with fewer elements can be used to reconstruct the original STS uniquely, every participant most share the piece of information to retrieve the secret.

The main object of study of the present work also lies under the context of cryptographic applications, specifically, studying STSs to construct authentication and identification protocols using interactive and zero-knowledge techniques. These protocols are based on the isomorphism problem on STSs, which, in turn, is strongly related to graph theory, since it has been proved that the isomorphism problem between combinatorial designs is polynomially equivalent to the graph isomorphism problem [11]. In fact, some of the hardest instances in graph theory arise from combinatorial structures [43], such as strongly regular graphs, block designs, and coherent configurations. In this context is that we study Steiner triple systems as a source of difficult instances, suitable for cryptographic purposes. A closer inspection of such combinatorial objects allows defining further transformations that will be helpful in to study the construction of a ZKP proof system [48], which will be detailed in 6.3. As far as we know, STSs have not been used before for identification purposes.

1.2 Objectives

The main goal of this work is to propose new cryptographic tools based on difficult combinatorial problems related to Steiner triple systems and identify instances that are suitable for cryptographic applications. In this respect, the investigation developed has considered two possible outcomes: either the isomorphism problem admits the definition of difficult instances, or the problem is not hard enough and can be solved efficiently. The examination of this problem with existing techniques leads to assume the former sentence as truth up to now.

The specific goals to achieve the main objectives are:

- Implement and improve known algorithms to solve the isomorphism problem on Steiner triple systems.
- Characterize difficult instances of the problem. This step is a consequence of the improvement of the existing isomorphism solvers.

- Provide theoretical and experimental evidence of the difficulty of the proposed instances.
- Design authentication and identification protocols based on zero-knowledge proof systems using the information gathered from the previous objectives.

1.3 Research products

We can enumerate the participation in conferences and scientific publications product of the results of the thesis work.

Conferences:

- **Steiner Triple Systems and Zero Knowledge Protocols**. Sagols Troncoso, F.; Morales-Luna, G.; González Fernández, E. XV Reunión Española sobre Criptología y Seguridad de la Información, Granada, Spain, October 3–5, 2018.
- **Public Key Infrastructure based on multivariate cryptography.** González Fernández, E., Morales-Luna, G., Sagols Troncoso, F., García Villalba, L.J. IV Jornadas Nacionales de Investigación en Ciberseguridad, Donostia-San Sebastián, Spain, June 13–15.
- **Infraestructura de Clave Pública en la Industria de Pagos con Tarjetas de Crédito**. González Fernández, E.; Morales-Luna, G.; Sagols Troncoso, F. Décimo Segundo Coloquio Nacional de Códigos, Criptografía y Áreas Relacionadas (CNCCAR), June 26–28, 2017
- **Protocolos de conocimiento nulo basados en problemas de geometría algebraica**. González Fernández, E.; Sagols Troncoso, F.; Morales-Luna, G. XLIX National Congress of the Mexican Mathematical Society (SMM), Aguascalientes, October 23–28, 2016.
- **Procedimientos de autenticación de conocimiento nulo mediante técnicas de geometría algebraica**. González Fernández, E.; Morales-Luna, G.; Sagols Troncoso, F. II Jornadas Nacionales de Investigación en Ciberseguridad, Granada, Spain, June 15–17, 2016

Publications:

- González Fernández, E., Morales-Luna, G., Sagols Troncoso, F. Zero Knowledge Authentication Protocols with Algebraic Geometry Techniques, *Appl. Sci.* **2020**, 10(2), 465.

1.4 Outline

The document is structured as follows. In Chapter 2 some basic concepts related with Steiner triple systems are briefly introduced. This chapter settles most of the notation that will be used through all the work. Additionally, a more detailed presentation of the early results briefly introduced in this chapter will be provided: the implementation of cryptographic schemes based on the $\mathcal{MQ}$ problem and a zero-knowledge protocol based on the graph isomorphism and $\mathcal{MQ}$.

In Chapter 3, more advanced topics regarding STSs are discussed. This discussion will be fundamental to understand the improvements of the isomorphism techniques and the construction of difficult instances. Additional transformations which produce non-isomorphic systems are examined for further applications.

Subsequently, in Chapter 4 the Miller's technique is studied. Up to now, this is the best algorithm known to solve the isomorphism problem for STSs, and more generally, for quasigroups. Improvements for some difficult instances are proposed. The analysis of the possible improvements allows to perform a more accurate description and estimation of the hardness of the isomorphism problem. In effect, this chapter turns out to be the most relevant from the combinatorial point of view. Some algorithms are developed and compared, and experimental evidence showing improvements of the Miller algorithm are exposed, though the asymptotical complexity is not reduced.

In Chapter 5, all previous theory studied and developed is used to achieve one of the main goals of this work: the characterization and construction of difficult instances, suitable for cryptographic purposes. A family of systems will be found to be resistant against the original Miller algorithm but not against the proposed improvements. Another group of systems will provide resistance against both techniques, and will be considered as the base for the authentication protocols.

The definition of zero-knowledge proof systems is carried out in Chapter 6. The first and second are an adaptation of the classical isomorphism and non-isomorphism interactive proofs defined for the graph isomorphism problem. The third protocol adds a more complex problem for the initial public and private information: the problem of inverting a composition of switching transformations. Since these transformations are

believed to connect every isomorphism class, this conjecture leads us to think that the problem is harder than the isomorphism problem, considering that an isomorphism can be expressed as a suitable sequence of transformations.

Finally, in Chapter 7 the conclusions and future directions of the current research are explored.

Chapter 2

Preliminaries and early work

In this section the basic background and notation of the mathematical objects which will be used frequently along this work are introduced. Some related contributions which are part of early research stages will be briefly discussed. These contributions are part of some original topics considered before establishing the main subject of study: the isomorphism problem in Steiner triple systems.

2.1 Graph theory

A *graph* is a pair $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is a set on n elements, the *vertices*, and E is a subset of $\binom{V}{2} := \{e \subset V \mid |e| = 2\}$, the *edges*. The *order* and *size* of G are the cardinality of the sets V and E, respectively. Two different vertices $v_1, v_2 \in V$ are *adjacent* if they are connected by an edge. Analogously, two different edges $e_1, e_2 \in E$ are *adjacent* if they share one and only one vertex. The graph $\overline{G} = (V, \overline{E})$ defined by $\overline{E} = \{v_i v_j \in \binom{V}{2} \mid v_i v_j \notin E\}$ is the *complementary graph* of G. This consists of pairs of non-adjacent vertices.

If two disjoint subsets $V_1, V_2 \subset V$ exist such that $V_1 \bigcup V_2 = V$ and such that every edge has vertices in both sets V_1 and V_2, then the graph is said to be *bipartite*. Furthermore, G is *complete bipartite* provided that every vertex in V_1 is connected to every vertex in V_2 and vice versa.

Now, consider two graphs $G = (U, D)$ and $H = (V, E)$. Consider a bijections of sets $\phi : U \to V$ that preserves edges, i.e., if $\{u, v\} \in D$ implies $\{\phi(u), \phi(v)\} \in E$. The ϕ is an *isomorphism* between G and H, and G and H are said to be *isomorphic*, denoted $G \approx H$. The graph isomorphism problem (GI) is defined as the task of finding an isomorphism between G and H, or deciding that this they are not isomorphic. Formally, GI can be defined as follows.

Decision problem

Instance: Two graphs $G = (U, D), H = (V, E)$.

Solution: $\begin{cases} 1 & \text{if there is an isomorphism } \phi : G \to H \\ 0 & \text{otherwise.} \end{cases}$

Search problem

Instance: Two graphs $G = (U, D), H = (V, E)$.

Solution: Either a proof that H and G are not isomorphic or the isomorphism $\phi : G \to H$.

Finally, a *matching* in a graph G is a subset $M \subseteq E$ with the property that no two edges $e_1, e_2 \in M$ are adjacent. The matching is *perfect* if, in addition, every vertex of G is an paired by an edge of M.

2.2 Interactive and zero-knowledge proof systems

Some very handy cryptographic tools used for authentication and identification services are the zero-knowledge proofs. A basic description of such systems consists of two parties: the *verifier* performs a series of questions to the *prover*, who must answer correctly in each round to convince the verifier. The prover will be capable of answering correctly on each round only if he has legitimate information.

For this process to be securely implemented, some characteristics regarding the interaction of the involved parties are desirable. The whole verification process should be computationally efficient for an authentic verifier, whereas it must be infeasible for an unauthentic prover to impersonate the authentic one. Furthermore, no information that allows a malicious verifier to reveal the prover's secret can be gathered, though this is commonly relaxed to "no statistically significant information". The following points summarize the desirable characteristics of a ZKP system:

- *Completeness.* An authentic prover will always be accepted by an honest verifier.
- *Soundness.* On interacting with a non-authentic prover, the verifier will reject with a very high probability.
- *Zero-knowledge.* A malicious verifier is not capable of getting any extra information from the challenge-response procedure, other than the correctness of the assertion.

This means that a verifier will always accept an authentic prover. However, a malicious prover has a chance to impersonate an authentic one, but with very small probability.

2.3 Quasigroups

Definition 1. A *quasigroup* is a set Q together with an operation $*$ such that:

i) Q is closed under $*$.

ii) For every $a, b \in Q$ there exist unique solutions $x, y \in Q$ such that $x * a = b$ and $a * y = b$.

An *isomorphism* between two quasigroups $(Q_1, *), (Q_2, \circ)$ is a bijection $\phi : Q_1 \to Q_2$ such that $\phi(x * y) = \phi(x) \circ \phi(y)$ for every $x, y \in Q_1$.

For simplicity, we refer to the pair $(Q, *)$ defining a quasigroup by Q, and the operation $x * y$ by xy. Exponential notation x^n is also preferred. A subset $H \subset Q$ is a *subquasigroup* if H is a quasigroup by itself. Let J be a subset of Q. The *subquasigroup generated by* Q, denoted $\langle J \rangle$, is the smallest subquasigroup containing J. If $\langle J \rangle = Q$ then J is a set of *generators* for Q. An isomorphism of quasigroups $\phi : Q_1 \to Q_2$ can be completely defined by a suitable set of generators $\{x_i, \ldots, x_m\} \subset Q_1$, and $\{y_1, \ldots, y_m\} \subset Q_2$, where $y_i = \phi(x_i)$ for $i = 1, \ldots, m$. It is worth observing that not every minimal generating set has the same size, for instance, consider the symmetric group on n symbols $\mathbf{S}_n$. It is well-known that a generating set of minimum size consists of 2 elements: a n-cycle and any transposition. However, another minimal generating set exists, consisting of $n - 1$ adjacent transpositions. For $n > 3$ the symmetric group admits minimal generating sets of different size.

A quasigroup that satisfies $x^2 = x, x(xy) = y$, and $(yx)x = y$ for any two elements $x, y \in Q$ is known as a *Steiner Quasigroup*. The nomenclature of such quasigroups will become clear once we address the theory of Steiner triple systems and its relation to quasigroups with more detail. Further results on quasigroups in general can be found in [47, 51].

2.4 Steiner triple systems

Definition 2. A *Steiner triple system* (STS) of *order* n, denoted STS(n), consists of a pair $S = (X, \mathcal{B})$ where X is a set of n elements, the *symbols* or *points*; and $\mathcal{B}$

is a set of unordered triples $\{x, y, z\} \subset X$, the *blocks* or *lines*, such that for any pair $\{x, y\} \subset X$ there exists a unique $B \in \mathcal{B}$ with $\{x, y\} \subseteq B$.

Standard counting arguments prove that each element in X must occur in exactly $r = \frac{n-1}{2}$ blocks, and that the triple system consists of exactly $b = \frac{n(n-1)}{6}$ blocks. Since both r and b are integers, we get necessary conditions for the existence of an STS(n), namely, $n \mod 6 \in \{1, 3\}$, which, in fact, turn out to be sufficient.

Two STS $S = (X, \mathcal{B})$ and $S' = (X', \mathcal{B}')$ are *isomorphic* if there is a permutation $\phi : X \to X'$ such that $\mathcal{B}' = \{\{\phi(i), \phi(j), \phi(k)\} | \ \{i, j, k\} \in \mathcal{B}\}$. The permutation ϕ defines a permutation of blocks, called an *isomorphism* between STSs. Since the permutation and the isomorphism are highly related, the isomorphism will also be denoted ϕ. For simplicity, the set of points X will be regarded as the set of integers $\{0, \ldots, n-1\}$, unless another definition is specified.

The smallest example of a STS is the system composed by a single block, namely, $S = (\{0, 1, 2\}, \{\{0, 1, 2\}\})$. However, we may also think of the empty set as a degenerate STS $S = (\emptyset, \{\emptyset\})$. For consistency with the requirements for the number of elements and blocks, we will not consider this case, and every STS will be nonempty.

Example 1. Let us illustrate the aforementioned definitions with a small example: a STS(7). The size of the set of blocks is $\frac{7 \cdot 6}{6} = 7$, and in this example, it is defined by

$$\mathcal{B} = \{\{0, 1, 2\}, \{0, 3, 4\}, \{0, 5, 6\}, \{1, 3, 5\}, \{1, 4, 6\}, \{2, 3, 6\}, \{2, 4, 5\}\}.$$

Two different representations of the set of blocks of can be seen in Figure 2.1: an edge-disjoint partition of the complete graph K_7 into triangles (left), and the set of lines of the Fano plane (right). In the graph representation, it is possible to see a strong relation of the isomorphism problem for STS and the isomorphism problem for graphs.

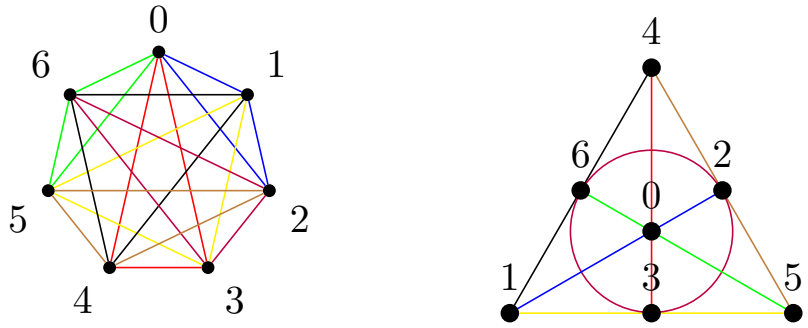


Figure 2.1: Representations of a STS(7)

A more systematic study of STSs has been carried out since Kirkman proposed the *Kirkman's schoolgirl problem*, named after him. The following is a transcription of the original text found in [32]:

> *Fifteen young ladies in a school walk out three abreast for seven days in succession: it is required to arrange them daily so that no two shall walk twice abreast.*

The solution to this problem is, in fact, a STS(15), but not every STS(15) holds the condition. To observe how quickly the problem grows in complexity, a solution to Kirkman's problem is pictured in Figure 2.2. In this case the number of blocks required to define the STS(15) is 35.

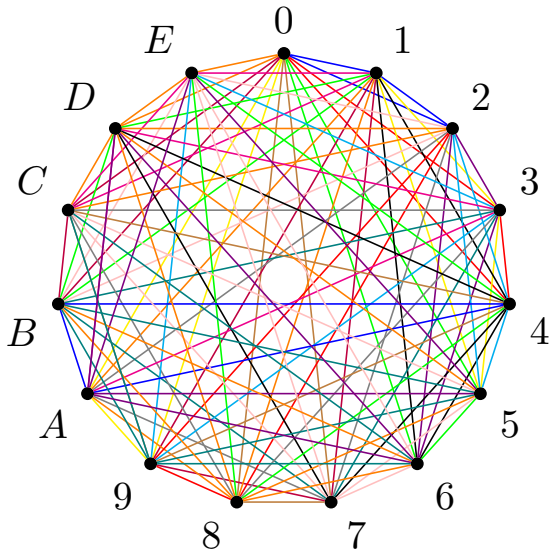


Figure 2.2: A solution to Kirkman's problem

In addition to the graph related representation, Steiner triple systems have a nice geometric interpretation, which will be useful throughout this work.

Definition 3. A Steiner triple system is a pair $(X, \mathcal{B})$ where X is a set of n elements called *points*, and $\mathcal{B}$ is a set of *lines*, satisfying:

- Every line is a 3-subset of X,
- Any two different points lie in exactly one line.

It is easy to see that both definitions are equivalent.

Several constructions for STSs exist in the literature, such as the methods proposed by Bose and Skolem [35], or iterative algorithms such as the Hill-climbing technique [53]. Also, the geometric representation can be useful when developing some constructions of STSs, such as the Fano plane or the Hall triple systems, introduced in Chapter 5.

The problem of deciding whether two STSs are isomorphic leads to an interesting classification problem: counting the number of non-isomorphic systems of a given order n. Two isomorphic STSs are said to belong to the same isomorphism class. The number of isomorphism classes explodes in a supra-exponential way as the cardinality of the base set increases. For orders 7, 9, 13, 15, and 19 the number of classes are 1, 1, 2, 80, and 11 084 874 829, respectively. The exact number of isomorphism classes for orders 21 and 25 is unknown. However, it is believed to be on the order of 10^{13} and 10^{15}, respectively. A survey of basic results on STSs can be found in [7, 35].

2.5 Early work

In this section, two main directions explored in the early stage of the definition of the desired research results are briefly explained. The discussion does not affect the understanding of the principal results introduced in subsequent chapters and can be skipped.

2.5.1 Implementation of cryptographic schemes based on multivariate cryptography

This section presents work developed in collaboration with MSc. Eliver Prez Villegas, former student of the Master program of the Department of Computer Science at CINVESTAV. The result of this effort can be found in `https://github.com/cinvestavcs-mexicocity/MQCrypto`. This consists of an implementation of some popular cryptographic primitives based on multivariate polynomials. It was later presented in the 2018 Spanish Cybersecurity Research Conference (JNIC2018).

Multivariate Cryptography is based on the proved difficulty of solving a multivariate quadratic system, known as the $\mathcal{MQ}$ problem, which belongs to the class of NP-complete problems [49]. Besides, $\mathcal{MQ}$ is harder than many collision finding problems, in particular, it is quantum resistant. We denote by $\mathbb{F}_q$ the finite field of $q = p^r$ elements, with p a prime number and r a positive integer, and consider positive integers m and n. Most of the PKC systems based on $\mathcal{MQ}$ consist of the following:

- an easily solvable quadratic map $Q : \mathbb{F}_q^n \to \mathbb{F}_q^m$,
- affine bijective maps $S : \mathbb{F}_q^n \to \mathbb{F}_q^n$, $T : \mathbb{F}_q^m \to \mathbb{F}_q^m$,
- a quadratic system P obtained as $P := T \circ Q \circ S$.

The system P is used as the public key and must be difficult to solve, while Q, S and T conform the private key. The main purpose of S and T is to hide the algebraic structure that makes Q easy to solve. An early attempt to develop a multivariate cryptosystem was published in [37], producing the *Matsumoto-Imai cryptosystem*. Since then, many other schemes have been developed by changing the way the central map Q is generated, being the most promising variations of the *Hidden Field Equations* (HFE) [44] and *Rainbow* [14] constructions.

In addition to quantum resistance, multivariate cryptosystems have attractive characteristics that makes them a suitable option for its implementation, e.g., fast computation, short signatures, diversity of constructions and natural resistance against timing attacks. On the other hand, many of the aforementioned schemes have shown vulnerabilities against cryptographic analysis. Even more, these cryptosystems produce very big keys compared to ECDSA and RSA. Though this is not a big restriction for devices with high memory capabilities, as servers, personal computers or mobile devices, it could be difficult to store keys in limited memory devices used in financial transactions, such as smart cards.

We have implemented the `mqcrypto` command-line application (available online: `https://github.com/cinvestavcs-mexicocity/MQCrypto`) to provide cryptographic primitives based on $\mathcal{MQ}$ and we expect to produce software to provide the full capabilities of PKI as a completion of the current work. The algorithms provided are a compilation of multivariate algorithms that have resisted cryptanalysis and are being considered by NIST to be part of the post-quantum public key standards.

2.5.1.1 Supported Algorithms

Up to now, we have integrated the following algorithms: Rainbow5640, Rainbow6440, Rainbow16242020, Rainbow256181212, Pflash, 3ICP, TTS6440 whose implementations and specifications can be found on `https://bench.cr.yp.to/ebats.html`. Additionally, we have implemented Sflash v1, Sflash v2 and UOV using the open-source software `SageMath`, available on `http://www.sagemath.org/`. This software is required if these algorithms are to be used.

The main cryptographic functions provided by the `mqcrypto` tool are described next:

- **Key generation**. Key pairs are created at random following the standard constructions, as can be seen in [13]. The user is asked for a password to generate an AES ciphering key which will be used to store the private key. Public and private keys are stored separately and the private one is never delivered in clear.

- **Signing**. The signature utility has as input a document, file or text, the ciphered private key, and the password used in the generation process. Also a digest method can be selected from Sha-256 and Sha-512. If no method is provided, Sha-256 is used by default. The output is either printed on the terminal or stored in a file if a path is provided.

- **Verifying**. This command verifies a signature using the corresponding public key. The inputs for this process are the signature, the public key, and the document or text that has been signed. Once it completes the execution, the command returns a positive answer if the sign is authentic and a negative one in other case.

In the following example the input file.txt will be signed and the signature will be stored in the file.sgn.

```
> mqcrypto sign Rainbow5640Private.pem -in file.txt
  -out file.sgn -sha256 -passin pass:password
```

The output is a Base64 string encoding an ASN.1 structured object. Codification of the key pair is exhibited with more detail in Section 2.5.1.2.

For a more detailed presentation of this software, we invite the reader to look at [45] and the additional documentation found on the Github page of the project.

2.5.1.2 Codification of Signatures and Keys

Resulting key pairs and message signatures follow an ASN.1 encoding. A public key consists of a set of m polynomials in n variables over a finite field. This information is encoded as follows

```
MPKCPublicKey ::= SEQUENCE {
    variables INTEGER, -- n
    polynomials INTEGER, -- m
    fieldchar INTEGER, -- p
    fielddegree INTEGER, -- k
    polynomialSet OCTET STRING, -- P}.
```

However, the values of n, m, p and k can be defined previously as domain parameters known to every entity involved in the authentication process. In this case, only the field `polynomialSet` must be provided. A private key is encoded as follows

```
MPKCPrivateKey ::= SEQUENCE {
    affine1 INTEGER, -- S
    affine2 INTEGER, -- T
    polynomialSet OCTET STRING, -- Q}.
```

In general, key length increases dramatically as m and n increase, since the growth of the number of coefficients per polynomial is quadratic. Nevertheless, some algorithms admit short codification for the private key by using numeric parameters. This is the case for schemes based on Matsumoto and Imai, where the parameter `theta` is sufficient to recover the private polynomial set.

```
MIPrivateKey ::= SEQUENCE {
    affine1 INTEGER, -- S
    affine2 INTEGER, -- T
    exponent INTEGER, --  theta}
```

This implementation aims to provide fundamental tools for employing standardized multivariate techniques for Public Key Infrastructure.

A brief analysis regarding possible applications of projective spaces in multivariate cryptography was carried out taking in account suggestions received in the predoctoral exam. In particular, techniques based on projective geometry lead to algorithms for searching solutions and estimating the number of points in an algebraic variety.

The number of points in an algebraic variety is important in schemes based on solving a set of polynomial equations. Too many of them might facilitate finding solutions which could be used to recover ciphered text or forge digital signatures. In this respect, the well-known Bezout's theorem [12, 8.7] provides an upper bound for the number of common solutions that can be found in algebraic curves. This could be of interest for desiging schemes with polynomials having total degree other than 2.

Other approaches aim to find the solutions by considering the geometric properties of some existing algorithms, such as the Euclid algorithm [28], or the *Relinearisation* and the XL algorithm [40], used in cryptology for multivariate equation solving. In the later work, authors describe different constructions of algorithm to find solutions to an equation system making use of the Veronese variety.

Due to the adjustment of the main objectives of this work, further research in this direction could not be further developed.

2.5.2 A zero-knowledge proof system based on an algebraic interpretation of the graph isomorphism problem

Now, a summary of the zero-knowledge proof system presented in [23] is introduced. The cited article can be consulted for futher details.

The notion of isomorphism between graphs can be translated to a strictly algebraic language. The idea is to perform a proper reduction from GI to $\mathcal{MQ}$ motivated by conventional reductions of several problems in graphs to boolean quadratic polynomials [21, 42]. For this, we need to consider a set of n^2 variables, denoted $\{X_{i,k}\}$ for $i, k = 1, \ldots, n$. The first set of polynomials to append, restrict any possible solution to values in the set $\{0, 1\}$. The polynomials are defined as follows:

$$X_{i,k}^2 - X_{i,k} \text{ for } i, k \in \{1, \ldots, n\}. \tag{2.1}$$

The next batch of polynomials restricts the zero-set to solutions that represent a perfect matching, i.e., exactly one vertex u_i from U is connected to a vertex of V and vice versa. This associates the solutions to the existence of a perfect matching M.

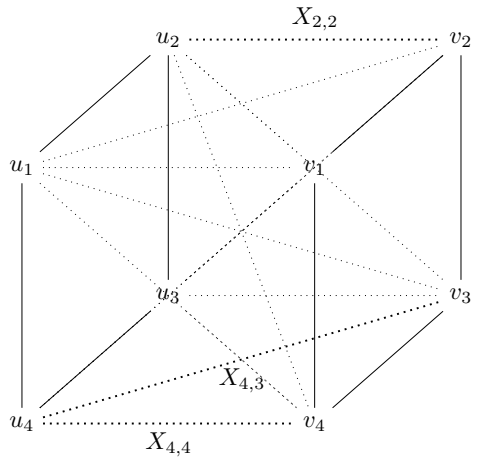
$$\begin{aligned} &\sum_{k=1}^{n} X_{i,k} - 1 \qquad \text{for } i = 1, \ldots, n \\ &\sum_{i=1}^{n} X_{i,k} - 1 \qquad \text{for } k = 1, \ldots, n \end{aligned} \tag{2.2}$$

The last set of polynomials guarantee that the solution is related exclusively to the isomorphism arising from the perfect matching:

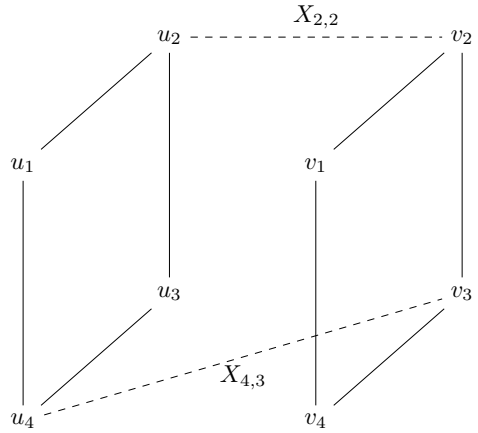
$$\begin{aligned} &X_{i,k}X_{j,l} \text{ for every } i, j, k, l \text{ which satisfy} \\ &\qquad (u_i u_j \notin D \wedge v_k v_l \in E) \vee \\ &\qquad (u_i u_j \in D \wedge v_k v_l \notin E) \end{aligned} \tag{2.3}$$

What has been explained can be observed in Figure 2.3.

The interaction process of the protocol starts by generating a pair of isomorphic graphs, its associated polynomial system F_0, and a solution $\mathbf{x_0}$ for F_0. Subsequently, a third graph K isomorphic to H is generated. Knowing the graph H and the applied permutation allows to obtain a second polynomial set F_1 and a its corresponding



(a) An isomorphism between G and H can be seen as a perfect matching in the graph $K_{U,V}$, preserving adjacencies between G and H.



(b) The edges u_2v_2 and u_3v_4 cannot belong simultaneously to M because $u_2u_3 \in D$, but $v_2v_4 \notin E$. The polynomial $X_{2,2}X_{3,4}$ is added to the ideal I.

Figure 2.3: Process to generate the polynomials set associated to the graph isomorphism.

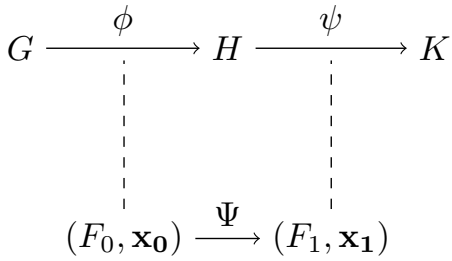


Figure 2.4: Isomorphism composition and resulting systems.

solution $\mathbf{x_1}$. The following diagram allows to visualize the operation here performed. Though the pair $(F_1, \mathbf{x_1})$ can be obtained in the same fashion than the pair $(F_0, \mathbf{x_0})$, i.e., by computing the polynomial set related to the corresponding graph isomorphism, a more direct approach consists in directly applying suitable permutations to the subindices k, l for the variables obtained from the edges of H and $\overline{H}$. In fact, let us define the permutation σ_ϕ by $\sigma_\phi(i) = k$ if $\phi(u_i) = v_l$. Then the edge $u_iu_j \in D$ transforms into edge

$$\phi(u_i)\phi(u_j) = v_{\sigma_\phi(i)}v_{\sigma_\phi(j)}.$$

A similar permutation σ_ψ, dependant on the action ψ, is obtained by relating edges of graph H and edges of graph K. The set of polynomials fulfilling condition (2.3) leads

to a direct definition of the set of polynomials corresponding to H and K obtained from the public polynomial set as

$$X_{\sigma_\phi(i),\sigma_\psi(k)} X_{\sigma_\phi(j),\sigma_\psi(l)}. \tag{2.4}$$

A solution for the system F_1 is provided by applying permutations σ_ϕ, σ_ψ to reorder the entries of the vector $\mathbf{x_1}$ in a similar fashion.

Observe that applying the permutation σ_ψ to the subindices of $X_{i,k}$ is equivalent to apply an affine transformation T, which might be represented by a matrix with one and only one element with value 1 on each column and each row (a permutation matrix) defined by

$$T(i,j) = \begin{cases} 1 & \text{if } j = \sigma_\psi(i) \\ 0 & \text{otherwise.} \end{cases}$$

A similar transformation S is related to ϕ, this time, it is applied on the right side.

$$S(i,j) = \begin{cases} 1 & \text{if } j = \sigma_\phi(i) \\ 0 & \text{otherwise.} \end{cases}$$

Indeed, S, T can be used to compute the new polynomial see as $\Psi(F_1) = S \circ F_0 \circ T$, and the new solution to such system by $\mathbf{x_1} = \Psi(\mathbf{x_0}) = S \cdot \mathbf{x_0} \cdot T$, which consists of a matrix multiplications.

Finally, if instead of using the isomorphism $\psi : H \to K$ to obtain the second polynomial system, the composition $\gamma = \psi \circ \phi$ is used, we get a third system, constructed by computing the new set $X_{i,\sigma_\gamma(k)} X_{j,\sigma_\gamma(l)}$, which requires of a single permutation, and in matrix notation only of the inner affine transformation T. Since both systems rely on the difficulty of computing a graph isomorphism, theoretically anyone could be used without losing security in the defined protocol.

2.5.2.1 Authentication protocol

The complete authentication protocol is outlined by the following steps, which are performed between Peggy (the prover) and Victor (the verifier):

Key Generation:

1. Peggy picks a graph G and randomly generates a permutation of the set $\{1, \ldots, n\}$. This permutation is used to create the isomorphic graph H together with its isomorphism ϕ. Then the public key F_0 using the technique aforementioned. The private key is pair $(F_0, \mathbf{x_0})$, the public system together with a solution.

Authentication:

1. Peggy generates a permutation σ for the set $\{1, \ldots, n\}$ at random and computes the polynomial system F_1, which is sent to Victor as a *compromise*.

2. Victor creates a challenge by selecting at random $b \in \{0, 1\}$. Victor sends b to Peggy.

3. Once Peggy has received b she must answer accordingly:

   - if $b = 0$, she sends the transformation Ψ to Victor.
   - if $b = 1$, then she sends the solution $\mathbf{x_1}$ of F_1.

4. According to the value of b Victor performs the following to authenticate Peggy:

   - if $b = 0$, he computes the system $F_1' = \Psi(F_0)$ and verifies whether he $F_1' = F_1$.
   - if $b = 1$, he checks whether $F_1(\mathbf{x_1}) = 0$ or not.

Since the problem relies in part on the hardness of the GI problem, this analysis led us to search suitable instances, which in turn guided the search to the combinatorial field, which is explored through the rest of this work.

Chapter 3

Steiner triple systems

The definition of STS and some basic information has already been discussed in Chapter 2. In this chapter, further topics related to STSs are introduced and discussed.

3.1 Subsystems and partial systems

Definition 4. A *partial Steiner triple system* (PSTS) P consists of a set X, the *points*, and a set triples $\mathcal{P} \subset \binom{X}{3}$, the *blocks*, such that every point lies in at least one block and any pair of different points lies in at most one block of $\mathcal{P}$. The *order* of P is $|X|$. A *subsystem* of a STS $S = (X, \mathcal{B})$ consists of the pair $T = (Y, \mathcal{C})$ with $Y \subset X$, $\mathcal{C} \subset \mathcal{B}$, and such that $(Y, \mathcal{C})$ is a STS.

With the above definition, a (PSTS) can be regarded as a STS with "missing" blocks. It is clear that subsystems of isomorphic STSs have a one-to-one correspondence, a result that can be useful when looking for isomorphisms between two systems. By defining the intersection between two subsystems $T_1 = (Y_1, \mathcal{C}_1), T_2 = (Y_2, \mathcal{C}_2)$ of S as $T_1 \bigcap T_2 = (Y, \mathcal{C})$ where

$$\mathcal{C} = \mathcal{C}_1 \bigcap \mathcal{C}_2, \quad Y = \bigcup_{B \in \mathcal{C}} B.$$

We get the following result.

Proposition 1. *Consider a STS S and two subsystems $T_1, T_2 \subset S$. Then $T_1 \bigcap T_2$ is either empty or a subsystem.*

Proof. Let $T_1 = (Y_1, \mathcal{C}_1)$ and $T = (Y_1, \mathcal{C}_1)$ be two subsystems of a STS $S = (X, \mathcal{B})$. Consider all triples $\mathcal{C} = \{B_1, \ldots, B_l\} \in \mathcal{C}_1 \bigcap \mathcal{C}_2$. Let $Y = \bigcup_{i=1}^{l} B_i$. if $|\mathcal{C}| = 0$ we are

done. Otherwise, consider any pair $\{x, y\} \subset Y$. From the definition of Y, it follows that at least one block B_i contains $\{x, y\}$. Now, from the definition of S, exactly one block contains it, as required. □

With this result, it is possible to argue that there is a unique subsystem that contains any subset of blocks, leading to the following definition.

Definition 5. Let S be a STS and $\{B_1, \ldots, B_m\} \subset \mathcal{B}$. The *subsystem generated by* $\{B_i\}_{i=1}^m$ (denoted $\langle B_1, \ldots, B_n \rangle$) is the smallest subsystem $\overline{T}$ containing $\{B_i\}_{i=1}^m$, and it is defined as

$$\overline{T} = \bigcap_{\substack{\{B_i\}_{i=1}^m \subset T \\ T \text{ is a subsystem}}} T.$$

In addition to the subsystem generated by a subset of lines, a subsystem generated by a subset of points $Y \subset X$ is the subsystem $(\overline{Y}, \overline{\mathcal{C}})$ of the smallest order with $Y \subset \overline{Y}$. In the subsequent, we will consider only non-trivial subsystems, i.e., subsystems with at least 7 elements. A STS without non-trivial subsystems is said to be *subsystem-free*.

In the same fashion as in vector spaces, it is of interest knowing the least quantity of points or lines needed to generate the full system. Though it is possible to deal with these questions with the items already defined, related topics developed in subsequent chapters will make this task easier.

3.1.1 Embedding systems and defining sets

The *embedding problem* in the context of STSs is a decision problem defined as follows:

Instance: A PSTS $P = (Y, \mathcal{P})$ and an admissible integer k.

Solution: $\begin{cases} 1 & \text{if there is an STS}(n) \text{ containing } P \\ 0 & \text{otherwise.} \end{cases}$

The Doyen-Wilson Theorem [15] establishes that a PSTS of order k can always be embedded into a STS(n) if $n \geq 2k + 1$ is an admissible integer. Later, Colbourn [6] proved that the problem is NP-complete for other values of n, specifically, for $n < 2k$. PSTS and subsystems are useful, considering the following applications:

- Considering a PSTS, the problem of extending to a full STS can be considered for cryptographic purposes.

- If the set pf blocks $\mathcal{P}$ of a PSTS P can be extended uniquely to the set of blocks $\mathcal{B}$ of a STS S, then a reduced representation of S can be provided, i.e., the partial system gives the same information than the full system.

Thus, this problem sets the ground for interesting applications in cryptography. It is also interesting in regard to the efficient codification of STSs, since a small subset that can be easily embedded uniquely in a STS will have the same amount of information than the full set. However, if not selected appropriately, several other systems can be obtained, or even worse, the problem of retrieving the original system can be intractable.

3.2 Cycle Graphs

The notion of cycle graphs and cycle lists, to be introduced next, were formalized in [52]. They provide useful invariants to deal with the problem of counting isomorphism classes. However, operations between systems can be derived from these objects. They will be utterly useful for the analysis of difficult instances performed in Chapter 5. This section starts by providing a formal definition of cycle graphs.

Definition 6. Let $S = (X, \mathcal{B})$ be an STS(n). Consider a pair $\{x, y\} \subset X$ and the unique block $B = \{x, y, z\}$ containing both, x and y. The *cycle graph* G_{xy} is defined by the set of vertices $V_{xy} = X - B$ and edges

$$E_{xy} = \{\{u, v\} \mid \{u, v, x\} \in \mathcal{B} - \{B\}\} \bigcup \{\{u, v\} \mid \{u, v, y\} \in \mathcal{B} - \{B\}\}.$$

It will be evident later why the graph associated to the pair $\{x, y\}$ is called cycle graph. We continue with an example to better understand the nature of these objects.

Example 2. Consider the STS(15) represented in Table 3.1. In this depiction, each column must be regarded as a block. This representation will be used in the rest of this work.

```
000000011111122222233334444555566666
13579bd3478bc3478bc789a789a789a789a
2468ace569ade65a9edbceddecbcdbeebdc
```

Table 3.1: A Steiner triple system of size 15.

Let us compute the cycle graph G_{01} using Definition 6. The vertex set is defined as $V_{01} = \{3,4,5,6,7,8,9,a,b,c,d,e\}$. Now, continue with the set of edges E_0 by

selecting every pair $\{a, b\} \subset V_{01}$ such that, together with 0, conforms a block in S. Then

$$E_0 = \{34,\ 56,\ 78,\ 9a,\ bc,\ de\}.$$

Similarly, $E_1 = \{35,\ 46,\ 79,\ 8a,\ bd,\ ce\}$. Following Definition 6, the cycle graph for the pair 01 is defined by the edges $E_{01} = E_0 \bigcup E_1$. The graph G_{01} is shown in Fig. 3.1.

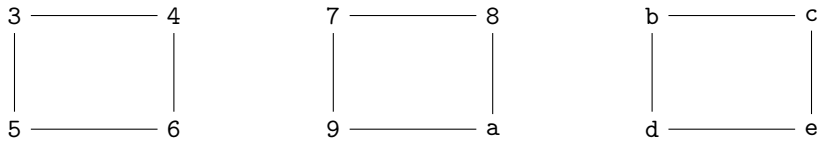


Figure 3.1: Cycle graph of the STS(15) example.

The resulting graph G_{01} consists of 3 disjoint cycles. Furthermore, each cycle is of even length. In fact, every cycle graph G_{xy} is 2-regular and bipartite, i.e., every graph consists of a disjoint union of even length cycles, as we show next.

Proposition 2. *Let $S = \{X, \mathcal{B}\}$ be an STS(n), and x and y two different elements of X. The cycle graph G_{xy} is a disjoint union of cycles of even length on $n-3$ vertices.*

Proof. From Definition 6, we have $V_{xy} = X - \{x, y, z\}$. Thus V_{xy} consists of $n-3$ vertices. Next, we show that G_{xy} is 2-regular, thus a union of disjoint cycles. Consider a vertex $u \in V$ and the blocks containing $\{u, x\}$ and $\{u, y\}$. These blocks are different, otherwise $u = z$, which is not possible from the definition of V_{xy}. Then, at least two different edges are incident on u, those arising from blocks $\{u, v, x\}$ and $\{u, w, y\}$. By definition, no other block contains $\{u, x\}$ or $\{u, y\}$, then no more edges in E_{xy} are incident on u, and G_{xy} is 2-regular.

Next, it is verified that the length of each cycle is even. It should be noticed that the edges of G_{xy} can be alternately labeled with x or y, thus having an even number of edges. □

Isomorphic STSs will have cycle graphs strongly related by the defition of an siomorphism between them.

Proposition 3. *Consider two isomorphic STSs S, S' with isomorphism ϕ. Now, let $G_{x,y}$ be a cycle graph of S corresponding to the pair $\{x, y\}$. Let $x' = \phi(x), y' = \phi(y)$ and the cycle graph $G'_{x'y'}$ of S' . Then ϕ defines a graph isomorphism between $G_{x,y}$ and $G'_{x',y'}$.*

Proof. It is clear from the definition that and edge $\{u, v\} \in G_{x,y}$ if and only if $\{\phi(u), \phi(v)\} \in G'_{x',y'}$. This provides an isomorphism between graphs. □

This leads to a straightforward definition of invariants that summarize better the structure of the graphs.

Definition 7. The *cycle list* associated to a cycle graph is the list of sizes of its individual components $(c_1, \ldots, c_l)$ with $c_i \geq c_{i+1}$. The set of all $\binom{n}{2}$ cycle lists is known as the *cycle structure* of S.

Example 3. Considering the system of Example 2, the cycle list for this cycle graph is given by $(4, 4, 4)$. An additional object related to the cycle graphs is that of *cycle vector*, which is a tuple that shows the distribution of the cycle lists. Consider the possible values of the cycle lists of S, we can enumerate them as $l_1 = (4, 4, 4), l_2 = (8, 4), l_3 = (6, 6)$ and $l_4 = (12)$. In the STS of Table 3.1 the resulting cycle vector is (21,36,0,48), meaning that it contains 21 graphs with cycle list l_1, 36 with cycle list l_2 and so on.

All of the aforementioned invariants can be calculated efficiently, being the complete cycle structure, which is computed in time and space $O(V^3)$, the most computationally expensive. However, non-isomorhic systems can share the same invariants. Take, for example, the triple systems #7 and #13 presented in the canonical list of 80 non-isomorphic STS(15), shown in [7] Table 1.28. Both of these systems have cycle vectors (21,36,0,48), even though they are not isomorphic.

A more simple invariant consists in considering only the number of cycles of length 4. In fact, the so-known *Pasch configurations* or *quadrilaterals*, closely related to these cycles of minimal length, have been more carefully studied. A quadrilateral consists of a set of 4 blocks in $\mathcal{B}$ of the form

$$Q = \{(a, b, c), (a, d, e), (b, d, f), (c, e, f)\}$$

whenever they exist. Three different cycles of length 4 can be obtained from this quadrilateral, corresponding to the cycle graphs G_{af}, G_{be}, and G_{cd} correspondingly. The aforementioned cycles can be seen in Fig. 3.2.

Then, a STS containing M different quadrilaterals will contain $3M$ different cycles of length 4. Some results about the number of quadrilaterals that can be contained in STSs can be found in [55]. These will be useful in the analysis of the ZKP proposed in Section 6.3. For this reason, we proceed to enunciate a couple of them.

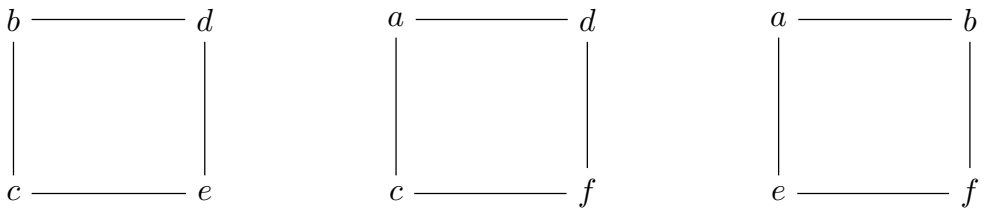


Figure 3.2: Cycles of length 4 obtained from a quadrilateral.

Theorem 1. *Let MQ(n) be the maximum number of quadrilaterals in any STS(n). Then*

$$MQ(n) \leq n(n-1)(n-3)/24.$$

Theorem 2. *An STS(n) contains exactly $n(n-1)(n-3)/24$ quadrilaterals if and only if it is isomorphic to a projective geometry PG(k, 2), for some $k \geq 2$.*

The projective plane $\mathrm{PG}(k,2)$ is built by considering $S = \mathbb{F}_2^k - \{0\}$. Lines consist of three points $x, y, z \in X$ satisfying $x + y + z = 0$. Each line will be considered as a block for the STS. Thus, to create a STS(n) with all cycle switches of size 4 the size of S must be $v = 2^k - 1$ for some k. Finally, as a consequence of Theorem 2, there is a unique STS, up to isomorphism, such that every cycle switch is of length 4.

For the most simple case, consider $k = 3$ and $n = 7$. We regard each vector in $\mathbb{F}_2^k$ as a binary representation of an integer. Then the STS is built on set $X =$ {1,2,3,4,5,6,7} and triples

$$\{123,\ 145,\ 167,\ 246,\ 257,\ 347,\ 356\}.$$

This example corresponds to the Fano Plane.

In addition to isomorphisms, more transformations between systems can be defined. In the next section, new operations on STSs which create systems that are not always isomorphic will be introduced.

3.2.1 The switching transformation

Let S be a STS, x, y two different elements of its base set, and consider the cycle graph G_{xy}. Since G_{xy} is a disjoint union of cycles, any vertex $v \in V_{xy}$ determines a component of the cycle graph uniquely. We denote the component containing v by G_{xy}^v. From the definition of cycle component, we deduce that $\{x, y, v\}$ is not a block of S.

Definition 8. Let $S = (X, \mathcal{B})$ be a STSs and $\{x, y, v\}$ a triple not contained in the set of blocks of S. Consider the component $C = G^v_{xy}$ and define the sets $\mathcal{T}$ and $\mathcal{T}'$ as follows:

$$\mathcal{T} = \{\{v_1, v_2, c\} \in \mathcal{B} \mid \{v_1, v_2\} \in E(C))\}, \quad \mathcal{T}' = \{B \triangle \{x, y\} \mid B \in \mathcal{T}\},$$

where $\triangle$ denotes the usual symmetric difference of sets. The *switching transformation* or *switch* under the cycle G^v_{xy} is the pair $S' = (X, \mathcal{B}')$ where

$$\mathcal{B}' = (\mathcal{B} - \mathcal{T}) \bigcup \mathcal{T}'.$$

We denote the transformation by $sw_C(S)$ or $sw^v_{xy}(S)$ interchangeably.

In other words, a switch consists of a replacement of the blocks in $\mathcal{T}$ of the form $\{v_1, v_2, x\}$ by the block $\{v_1, v_2, y\}$ and conversely. The set $\mathcal{T}$ is determined by the selection of the triple $\{x, y, v\}$.

Example 4. Consider again S as the STS(15) presented in Table 3.1. The representation of S is written again next.

S: `000000011111122222233334444555566666`
`13579bd3478bc3478bc789a789a789a789a`
`2468ace569ade65a9edbceddecbcdbeebdc`

Consider the cycle graph $G^9_{5,6}$. To produce $S' = sw^9_{5,6}(S)$, we proceed by computing the set of vertices and edges $V_{5,6}$ and $E_{5,6}$, respectively. Select the component of the cycle graph containing vertex 9 and compute $\mathcal{T}$ and $\mathcal{T}'$, as it has been previously defined. Figure 3.3 is a visual representation of the operation. A block is comprised of two vertices and the edge label. The resulting system S' is isomorphic

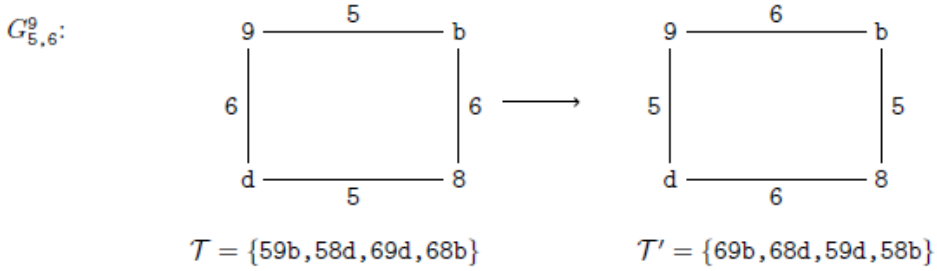


Figure 3.3: Visual representation of the cycle switch transform.

to system #2 in Table 1.28 of [7], the isomorphism being given by the transposition (d e). Now, by applying a new switching $S'' = sw^7_{5,6}(S')$ results again in a new STS, which, in this case, is exactly system #2. The first transformation takes S into a non-isomorphic system, but the second one transforms S' into an isomorphic one.

The result of a cycle switching is always a new STS of the same order. Both, isomorphic and non-isomorphic systems can be obtained by performing the switching operation.

Theorem 3. *Let $S = (X, \mathcal{B})$ be a STS(n) and $\{x, y, v\} \subset X$ a block not contained in $\mathcal{B}$. Then $S' = sw^v_{xy}(S)$ is an STS(n) which, in general, is not isomorphic to S.*

Proof. To verify this assertion, we will proceed in two steps: first, we show that the size of the block set of S' is exactly $\frac{n(n-1)}{6}$ and second, that every pair of elements $\{a, b\}$ is contained in a block of S.
1) Since $\mathcal{T} \bigcap \mathcal{T}' = \emptyset$ and $|\mathcal{T}| = |\mathcal{T}'|$, the number of blocks after applying sw^v_{xy} does not change.
2) Consider a pair of elements $a, b \in X$ and the block $B = \{a, b, c\} \in \mathcal{B}$ that contains it. Since block $\{a, b, c\} \notin \mathcal{T}$ is not replaced, it suffices to check pairs contained in blocks of $\mathcal{T}$.

- If $a, b \notin \{x, y\}$, then $c \in \{x, y\}$. If $c = x$ then the block B is transformed into $B' = \{a, b, y\} \in S'$. When $c = y$ the case is analogous.
- Now, consider $a = x$. Note that the block $\{y, b, c\}$, which contains $\{b, y\}$ will be transformed into $\{x, b, c\}$, then the pair $\{x, b\}$ lies in a block in $\mathcal{T}'$. The same analysis applies to $a = y$ and also to $b \in \{x, y\}$.

Since any pair is contained in at least one block and the number of blocks is $\frac{n(n-1)}{6}$, the resulting system S' is a STS. The fact that the transformation is not necessarily isomorphic, is followed directly by the result obtained in Example 2. □

By applying sw^v_{xy} when G^v_{xy} is a cycle of length $n-3$, the resulting STS is always isomorphic, with the isomorphism given by the permutation $(x\ y)$. A more general result can be stated.

Proposition 4. *Let S be a STS, $\{x, y\} \subset X$, and G_{xy} a cycle graph with connected components $C_1, \ldots, C_l$. Let S' be the STS*

$$S' = sw_{C_l} \circ \cdots \circ sw_{C_1}(S).$$

Then S' is isomorphic to S.

Proof. After applying switchings with all the cycles C_i, the result is equivalent to permute the set X with the transposition $(x\ y)$. □

Proposition 5. *Let S and C_i be as in the last proposition. Then*

$$sw_{C_i} \circ sw_{C_j}(S) = sw_{C_j} \circ sw_{C_i}(S)$$

for every $1 \leq i, j \leq l$.

Proof. For each cycle C_i, we consider the corresponding partial systems $\mathcal{T}_i = \{\{a, b, c\} \in \mathcal{B} \mid \{a, b\} \in C_i\}$ and $\mathcal{T}_i' = \{B \triangle \{x, y\} \mid B \in \mathcal{T}\}$. The result for $i = j$ is clear. If $i \neq j$ then $\mathcal{T}_i \bigcap \mathcal{T}_j = \emptyset$ and $\mathcal{T}_i' \bigcap \mathcal{T}_j' = \emptyset$. The set operations $(S - \mathcal{T}_i) \bigcup \mathcal{T}_i'$ are not dependant on the order in which they are performed, then the transformations are commutative. □

The switching operation is very useful to get new systems from previously known ones. In the case of STSs of order ≤ 19 it has been shown that any system can be obtained as a result of applying switching transformations [24, 30]. In other words, the isomorphism classes of systems of order at most 19 are connected with the switch transformation. The result is unknown for systems of higher order.

From Proposition 5 it can be concluded that the switch transformation is invertible, with an inverse being the transformation itself. Thus, an inverse of a composition of several switchings is just the composition of the applied transformation in reversed order.

An interesting task consists in finding a composition of switches that send a STS into another. Given the huge amount of systems for $n > 15$ and the infinite possible combinations of compositions, this problem is in fact a difficult one. The easiest case consists in inspecting a couple of systems transformed by applying a single switch. The required transformation can be easily found by looking at the blocks that differ from the block sets of S_1 and S_2.

Given the importance of the set $\mathcal{T}$ involved in the transformation, we will denote $\mathcal{T}_{xy}^v$ to the partial subsystem

$$\mathcal{T}_{xy}^v = \{\{a, b, c\} \in \mathcal{B} \mid \{a, b\} \in G_{xy}^v\}$$

to specify its relation to the switching operation defined by the graph G_{xy}^v.

3.2.2 Effect of switching on cycle graphs

In addition to the behavior of the switching transformation in STSs, it is also important to analyze its effect on cycle graphs. In the subsequent, G_{xy}^v will be the component of G_{xy} containing the vertex v, where $\{x, y, v\}$ is a suitable triple.

Note 1. *Consider a block $\{a, b, c\} \in \mathcal{B}$. The pair $\{a, b\}$ appears as an edge of $G_{c,x}$ for every $x \notin \{a, b\}$. This can be seen directly from the definition of $G_{c,x}$.*

The influence of the transformation can be easily observed by analyzing the edges affected. This is stated in the following proposition.

Proposition 6. *Let $C = G^v_{xy}$ and $\overline{C}$ its complement in G_{xy}. A cycle graph G_{ij} remains unchanged iff either*

1. *$\{i, j\} = \{x, y\}$, or*
2. *$\{i, j\} \subset V(\overline{C}) \bigcup \{z\}$, or*
3. *$|C| = 4$ and $\{i, j\}$ are opposite vertices in C.*

Proof. $\Rightarrow$)

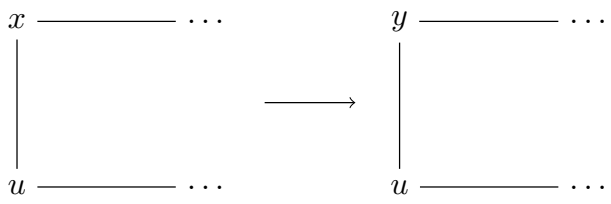
1. G^v_{xy} only switches vertices x, y when they appear in G_{ij}. None of these appear in V_{xy}.
2. Consider the sets $\mathcal{T}^v_{xy}$ and $\mathcal{T}_{ij}$, corresponding to the blocks related to the cycle graph component G^v_{xy} and the cycle graph G_{ij}. A particular case of these block sets can be seen in Figure 3.3. It is clear that, if $\mathcal{T}^v_{xy} \bigcap \mathcal{T}_{ij} = \emptyset$, then the structure of the cycle graph is not affected. That the intersection is empty means that no block with a pair of the form $\{x, v\}$ or $\{y, v\}$ with $v \in V(C)$ is present in $\mathcal{T}_{ij}$. Since x and y will be elements in $V(G_{ij})$, then we should have $\{i, j\} \subset V(\overline{C}) \bigcup \{z\}$, as stated.
3. If $|C| = 4$, then $\mathcal{T}^i_{xy}$ is a Pasch configuration, which can be regarded as

$$\mathcal{T}^i_{xy} = \{\{i_0, i_1, x\}, \{i_1, i_2, y\}, \{i_2, i_3, x\}, \{i_3, i_0, y\}\}$$

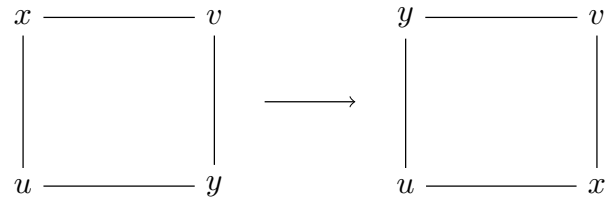
with $i = i_0, j = i_2$, since they are opposed vertices. Observe that G_{ij} has the cycle (i_1, x, i_3, y), which, after swapping x and y, is mapped to (i_1, y, i_3, x). Though relabelled, the obtained cycle is identical, but in reverse order. The remaining components of G_{ij} are not affected since their vertices are not elements of C.

$\Leftarrow$) We need to show that for any pair $\{i, j\}$, which is not identified by one of the three groups listed above, its cycle graph G_{ij} is modified.

First, note that $V(C) \bigcup V(\overline{C}) \bigcup \{z\} = X - \{x, y\}$. Let us start with $i \in V(C)$ and $j \neq x$. Then $x \in V(G_{ij})$ and the block $\{i, x, u\} \in \mathcal{T}$. Consider two cases:



(a) G_{ij} for $i \in V(C), j \notin \{x, y\}$



(b) $x, y \in V_{ij}$ as opposite vertices.

Figure 3.4: Effect of sw on cycle graphs.

1. $i \neq y$. Then after applying sw_C the edge $\{x, u\} \in E_{ij}$ is transformed into $\{y, u\}$.

2. $i = y$. Then the cycle G_{ij} is transformed into G_{ix}.

In any case, the cycle graph is transformed. Observe that the only case where this switch does not change the graph is for the case $|C| = 4$ and i, j are opposite vertices. □

Let us continue with a final observation regarding the switching operation. Consider a cycle graph G_{xy} with at least two components. Let v_i and v_j two vertices of G_{xy} from different components, with respective edges $\{v_i, v_{i+1}\}$ and $\{v_j, v_{j+1}\}$. Now, consider the transformation $sw^x_{v_i,v_{j+1}}$. This transformation maps the triple $\{v_i, v_{i+1}, x\}$ to $\{v_{j+1}, v_{i+1}, x\}$. Additionally, the triple $\{v_j, v_{j+1}, x\}$ is mapped to $\{v_j, v_i, x\}$. The result is shown in Fig. 3.5. In particular, if $G^x_{v_i,v_j+1}$ is a Pasch configuration, then the cycles $G^{v_i}_{xy}$ and $G^{v_j}_{xy}$ will be joined in a unique cycle.

A similar observation can be done in the opposite direction. By applying the switch transformation $sw^x_{v_i,v_j}$ to a cycle having edges $\{v_i, v_j\}$ and $\{v_{i+1}, v_{j+1}\}$, the original cycle will be split into two different cycles of shorter length. These operations are better visualized in Figure 3.5.

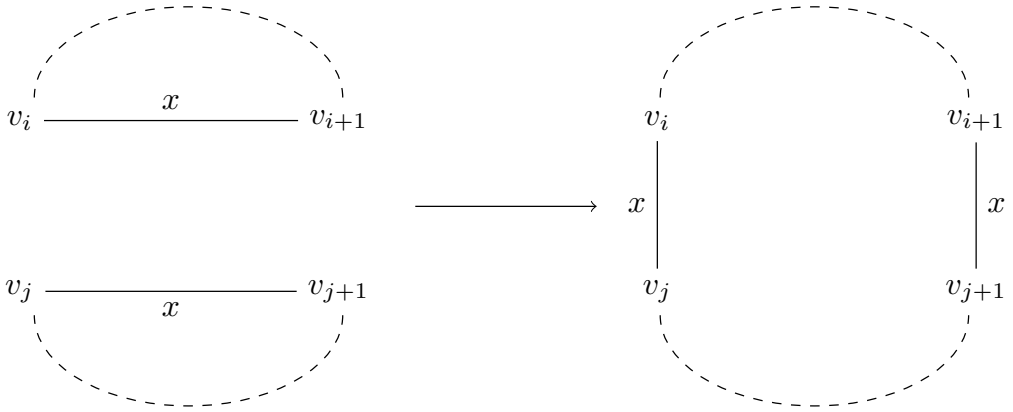


Figure 3.5: Joining two cycles with a switching transformation.

3.3 Configurations

Related to partial systems and cycle graphs, the concept of configuration has been widely studied since they provide additional characterizations of STSs that permit deciding isomorphism. This is, configurations provide more invariants to the existent catalog.

Definition 9. Let S be a STS. An *l-line configuration* is a subset of l lines (blocks) of the set $\mathcal{B}$. An l-line configuration on m points is a set of l blocks whose union has size m and is denoted as (m, l)-configuration.

In other words, a configuration consists of a partial system extracted from a STS. An example of the above definition is the Pasch configuration previously introduced. The existence problem arises again when trying to identify all possible l-configurations on m points. Isomorphisms between configurations are induced by STS isomorphisms. For instance, for $m = 6$ and $l = 4$, there is a unique configuration up to isomorphism, known as the Pasch configuration. A study of the number of possible configurations of small size can be found in [26]. We follow the notation there introduced.

Constructions of STSs avoiding certain configurations have been proposed, for instance, the anti-Pasch systems [25, 36], or the **6-sparse** systems, studied in [19], which are systems with no (8,6)-configurations.

The smallest anti-Pasch STS has order 15, numbered 80 in the Table 1.28 [6] and is unique (up to isomorphism) in this sense. Opposite to this behavior, the STS(15) number 1 has the maximum possible number of such configurations, likewise, unique up to isomorphism.

The set of triples involved in 6-cycles can be considered as an $(8, 6)$-configuration, which is isomorphic to $\{012, 034, 135, 246, 257, 367\}$. The 6-cycle configuration is one

of the five non-isomorphic $(8, 6)$-configuration that can be contained in Steiner triple systems. These configurations, denoted E_1, E_2, E_3, E_4 and E_5 in [26], together with the 2 $(7, 5)$-configurations D_1, D_2, known as *mitre* and *mia*, are essential for building difficult instances of the isomorphism problem, which will be addressed in Chapter 5. The following table extracted from [19], enumerates relevant $(n, n+2)$-configurations for $n \geq 6$.

n	Name	Blocks
4	Pasch	012, 034, 135, 245
5	Mitre	012, 034, 135, 236, 456
5	Mia	012, 034, 135, 245, 056
6	6-cycle	012, 034, 135, 246, 257, 367
6	Crown	012, 034, 135, 236, 147, 567
6		012, 034, 135, 236, 146, 057
6		012, 034, 135, 236, 146, 247
6		012, 034, 135, 236, 147, 257

Table 3.2: Small configurations in Steiner triple systems.

Different from what is done in [19], we will aim our efforts in finding systems with as many (8,6)-configurations, specifically, 6-cycles. The reason for this will become apparent in Chapter 5.

3.4 Trades

Consider the sets $\mathcal{T}$ and $\mathcal{T}'$ defined in Theorem 3. By replacing the triples in $\mathcal{T}$ by those contained in $\mathcal{T}'$, we obtained a new STS. These sets of triples can be regarded as partial subsystems of S and S', respectively. What has been done in this case was the result of applying a switch, but this can be easily generalized to provide another operation on STSs.

Definition 10. A *trade* is a pair of configurations $\{\mathcal{T}, \mathcal{T}'\}$ where each pair $\{x, y\} \subset X$ appears in some block of $\mathcal{T}$ if and only if it also appears in a block of $\mathcal{T}'$.

This definition ensures that after *trading*, the resulting system $S = (S - \mathcal{T}) \bigcup \mathcal{T}'$ is also a STS. The smallest trade corresponds, in fact, to Pasch configurations. Further details about trades can be found in [18] where suitable configurations and the determination of non-isomorphic trades of small size are discussed. For our purposes, the following result is enough to address our main goal.

Proposition 7. *Let $T = (Y, \mathcal{T})$ be a subsystem of S. Define $T' = \phi(T)$ for the isomorphism induced by a permutation of the set Y. Then (T, T') is a trade.*

Proof. Since T is a subsystem, every pair $\{x, y\} \subset Y$ is contained in some block of T. The same happens for T', since it is a Steiner triple subsystem on the same set, fulfilling the definition of trade. □

Chapter 4

The Isomorphism Problem for Steiner Triple Systems

We remind that two STSs $S_1 = (X, \mathcal{B})$, $S_2 = (X', \mathcal{B}')$ are isomorphic if there is a bijection $\phi : X \to X'$ that induces a bijection between triple sets $\mathcal{B} \to \mathcal{B}'$. Though in a brute force search, one must look at the $n!$ possibilities that could result in an isomorphism, the problem is can be addressed more effectively. Our main goal in this chapter is to characterize difficult instances for the isomorphism problem to propose parameters for cryptographic purposes.

Two different problems are discussed:

1. Given two systems, decide whether they are isomorphic or not.

2. Given two isomorphic systems, find an isomorphism between them.

Up to now, the best algorithms to solve these problems in general work by construction, i.e., by specifying a bijection that specifies the required isomorphism is given or proving that no bijection exists.

4.1 Miller's Algorithm

It is possible to regard a STS $S = (X, \mathcal{T})$ as a quasigroup by defining the pair $(Q, *)$ where $Q = X$ and $* : Q \to Q$ is defined as follows:

- For every pair of different elements $x, y \in S$, set $x * y = z$ if $\{x, y, z\} \in \mathcal{T}$.

- $x * x = x$ for every $x \in X$.

It is straightforward to verify that the quasigroup described above is a Steiner quasigroup. The relevance of these quasigroups will become more evident in Chapter 4. The main goal in this section is to extend the existing relations between quasigroups and STSs.

The first procedure to be considered was presented by Miller in 1978 [39]. This algorithm finds isomorphisms by construction in time $O(n^{\log(n)})$, and is suitable for many algebraic structures. The algorithm was proposed to find isomorphisms between quasigroups. To introduce the aforementioned algorithm, first let us note that for a subquasigroup H of Q, if $x \in Q - H$ then $|xH| \geq 2|H|$, i.e., the subquasigroup generated by appending an element to H at least doubles its size. This observation is key for finding a generator set of the quasigroup, which us the main result that leads to Miller's algorithm, shown in Alg. 1.

Algorithm 1 Miller's Algorithm for Quasigroup Isomorphism

Require: Two quasigroups Q, Q'

Ensure: Isomorphism $\phi : Q \to Q'$ are isomorphic or **false**otherwise

1: Find a set of generators H of size m with $m \leq \log(n)$
2: **for** each possible subset $H' \subseteq Q'$ of size m **do**
3: **if** there is a bijection $\phi : H \mapsto H'$ inducing an isomorphism **then**
4: **return** ϕ
5: **return false**

This algorithm has a complexity $O(n^{\log n})$, as it will be analyzed next. First, there are $\binom{n}{m}$ different subsets of size m in Q, thus, the loop in step 2 iterates $\binom{n}{m}$ in the worst case. Step 3 should test all possible assignments of the m generators. There are $m!$ possible bijections. We can conclude that at most $\binom{n}{m}m! = n(n-1)\dots(n-m+1) < n^m = O(n^{\log n})$ are required to determine an isomorphism or decide that Q, Q' are not isomorphic.

Thus, in order to find hard instances for the isomorphism problem, we should find systems such that the smallest generator set of its Steiner quasigroup has as many elements as possible. The relation of generators and quasigroups is clear. The more generators a quasigroup has, the more subquasigroups it contains.

Proposition 8. *A STS S contains a nontrivial subsystem $T \subset S$ if and only if its associated quasigroup Q_S contains a nontrivial subquasigroup $Q_T \subset t_S$. Moreover, there is a bijective relation between the subquasigroups and subsystems.*

Proof. Consider a STS S and a subsystem $T \subset S$. Since T is a STS by itself, it has an associated quasigroup Q_T. Since every element of Q_T is an element of Q_S, then Q_T is

a subquasigroup. Additionally, if $T' \neq T$ is another subsystem, then the underlying set of T' is different to that of T. Thus $Q_{T'} \neq Q_T$. □

From the last result, we note that STSs with several subsystems are required for the isomorphism problem to be difficult if addressed with Miller's algorithm. Nevertheless, a big amount of subsystems or, equivalent, a quasigroup with several generators do not guarantee that the problem is difficult, and one of the main drawbacks of Miller's algorithms relies in the selection of possible bijections done for every possible subset of size m, performed in step 2. Instead, a more intelligent selection by choosing suitable generators improves considerably the performance of the algorithm for the most difficult cases. The enhancements will be introduced in Section 4.3.

The next step is to identify more attributes that help to identify convenient systems. The desirable characteristics will arise in the next section while discussing another approach to address the isomorphism, this time directly in Steiner triple systems.

4.2 Algorithm based on cycle graphs

As mentioned in Section 3.2, cycle graphs and cycle lists provide useful invariants for fast non-isomorphism tests. Though isomorphic STSs have identical cycle structure, the converse is not necessarily true. In this section, we deliver a deeper analysis of the cycle graph to provide another isomorphism test. The computational complexity of such an algorithm is better than Miller's algorithm in most of the cases. Even though the complexity is not improved for every case, the most important result in the following analysis is the provision of characteristics required for difficult isomorphisms instances.

Let us recall that an isomorphism $\phi : S \to S'$ between STSs yields a bijection between isomorphic cycle graphs $G_{xy} \mapsto G_{\phi(x)\phi(y)}$. Then, to provide an isomorphism between STSs, we can focus on solving the isomorphism problem in graphs. Specifically, an isomorphism between graphs consisting of disjoint cycles will be required. Once a suitable solution for two given cycle graphs is found (one for each system), the next step is to verify that the selected assignation provides, in fact, an isomorphism for S and S'.

Consider two cycles of length l. Then, a total of $2l$ different isomorphisms can be defined between them. They are obtained by shifting the vertices forward and backward. If a cycle graph consists of a single cycle, the problem of finding an isomorphism is easily solved. In fact, we need only to test the possible isomorphisms

between the selected cycle of S and the different cycle graphs of S' consisting of a unique cycle. The problem gets harder when we consider cycle graphs conformed by the union of several cycles.

Proposition 9. *Let G and G' be disjoint unions of k cycles of length l. Then there are $(2l)k!$ different isomorphism between G and G'.*

Proof. Each cycle from G can be mapped to any cycle in G'. We have $k!$ different options for defining the component-wise isomorphism. The conclusion is obtained by considering the observation that two cycles have $2l$ different isomorphisms. □

Corollary 1. *Let G and G' be disjoint unions of $k = k_1 + \cdots + k_r$ cycles of length $l_1, \ldots, l_r$ correspondingly. Then there are $\sum_i (2l_i)k_i!$ different isomorphism between G and G'.*

Even though there are several possible isomorphisms between cycle graphs, just a few of them (if any) develop in a valid isomorphism for the underlying systems. It is possible to avoid verifying every isomorphism by extending partial bijections obtained from one or more components. Once two isomorphic cycle graphs are chosen to start the isomorphism search, Algorithm 2 can be used to extend a bijection from a partial definition. This algorithm allows to fix values from assigned vertices in step

Algorithm 2 Extending a partial bijection

Require: Steiner Triple Systems S and S' and a partial bijection ϕ
Ensure: An extension of the partial bijection

```
1: for each pair {a, b} ⊂ X do
2:   if {a, b} is mapped under φ then
3:     a', b' ← φ(a), φ(a)
4:     Search the triple {a', b', c'} containing {a', b'}
5:     if c is mapped under φ then
6:       if c' ≠ φ(c) then
7:         return false                    # inconsistent assignment
8:     else
9:       φ(c) ← c'
10: return φ
```

9. Previously, in step 7, unsuitable assignations are detected, allowing to discard the partial bijection. This is done by verifying that $\phi(xy) = \phi(x)\phi(y)$ for every $x, y \in X$.

If Algorithm 2 fails to extend an isomorphism for STSs, a new isomorphism between cycle graphs must be tried. If no isomorphism can be found from the selected couple of graphs, a new pair must be inspected. We are ready to provide the full algorithm for isomorphism test based on cycle graphs. First, since longer cycles fix more elements than shorter ones, we search in S for the cycle graph with bigger components. Another approach is to select the one with the least components. Once a good cycle graph has been chosen, we proceed by trying to find a suitable bijection by trying on every possible isomorphism between cycle graphs. If every isomorphic graph of S' has been tried and no STS isomorphism is found, we can conclude that S and S' are not isomorphic. These ideas are summarized in Algorithm 3.

Algorithm 3 Finding isomorphisms with cycle graphs

Require: Steiner triple systems S and S' of the same order.

Ensure: Isomorphism if exists or **false** otherwise.

1: $\{x, y\} \leftarrow$ any pair with G_{xy} having minimal number of components
2: **for** each $\{x', y'\} \subset X'$ **do**
3: **for** each possible graph isomorphism $\phi : G_{xy} \to G'_{x'y'}$ **do**
4: $\phi \leftarrow \text{EXTEND}(\phi, S, S')$ *# use Algorithm 2*
5: **if** $\phi : S \to S'$ is a complete isomorphism **then**
6: **return** ϕ
7: return **false**

The algorithm can be improved by starting step 3 with the longest connected component and using Algorithm 2 to accelerate the analysis. In fact, the number of cycle graph isomorphisms to consider can be drastically reduced according to the following result.

Proposition 10. *Let S be a STS and G_{xy} a cycle graph with a connected component $C \subset G_{xy}$ with $|C| \geq \frac{n-2}{2}$. The set $\mathcal{T}$, as defined in the switching transformation, generates the system S.*

Proof. The number of elements in the partial STS $\mathcal{T}$ is at least $\frac{n+1}{2}$, given by $Y = V(C) \bigcup \{x, y\}$. Let Q_S be the Steiner quasigroup defined by S. Since $|Y| \geq \frac{|Q_S|}{2}$, then $\langle Y \rangle = Q_S$. The bijective relation between subsystems and subquasigroups leads to the result. □

With this result, it is only required to find a cycle with appropriate length, i.e., with length at least $\frac{n-3}{2}$, which happens in several cases. For instance, the smallest case when no cycle with length less than $n-3$ exists is for $n = 15$. Only 2 out of the

80 non-isomorphic classes have such behavior, corresponding to the STS #1 and #2 in the canonical table shown in [9]. Every cycle graph in STS #1 consists of three disjoint cycles of length four each. In STS #2, the count of cycles with lengths (8,4) is 48, and for cycles with lengths (4,4,4) is 57. In the remaining systems, cycles of length 12 can be obtained. In view of what has been mentioned before, only system #1 represents a challenge to solve the isomorphism problem.

As a matter of fact, of the 11,084,874,829 different classes of order 19, only 86,972,331 posses non-trivial subsystems [31]. This means that 99.2% of these instances posses cycles of length 16, and then the isomorphism problem is efficiently solved. The remaining classes contain either a subsystem of size 7 or of size 9 (or both). From these results, it is expected that only a very small number of the systems of any order provide some difficulty. Thus, for a huge number of instances, the expected running time will be polynomial. A single cycle will suffice to run the isomorphism test.

4.2.1 Complexity of the algorithm

The complexity analysis for the proposed algorithm is mainly based on Propositions 9 and 10. According to the aformentioned results, the worst case would appear when every cycle graph has the maximum number of components, this is $k = \left\lfloor \frac{n-3}{4} \right\rfloor$. The ideal case would happen when every cycle has length 4. With the notation introduced in Proposition 9, $l = 4$ the maximum number of isomorphisms can be estimated as

$$(2l)k! = 8 \cdot \left\lfloor \frac{n-3}{4} \right\rfloor ! = O(n^{\left\lfloor \frac{n-3}{4} \right\rfloor})$$

However, as mentioned before, by using Algorithm 2 in combination with Algorithm 3, a bijection can be extended without using all possible isomorphisms, which can reduce the running time of the algorithm, and when a cycle of length $l \geq \frac{n-3}{2}$ is found, the number of possible graph isomorphisms to consider is reduced to $2l$. Now, since the procedure is performed for each cycle graph, the number of iterations is (at most) $\frac{n(n-1)}{2}$, the total complexity would rise to

$$\frac{n(n-1)}{2}(2l)k! = 8\frac{n(n-1)}{2} \left\lfloor \frac{n-3}{4} \right\rfloor ! = O\left(n^{\left\lfloor \frac{n-3}{4} \right\rfloor + 2}\right)$$

This is much more expensive than the Miller's algorithm, but only for very few cases. In the vast majority of the systems, k is reduced to 1 and $l \geq n/2$. The performance of the algorithm is drastically improved to $O(n^3)$.

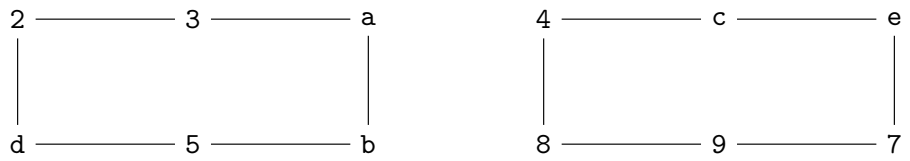
The upper bound for the worst case complexity of the proposed algorithm can be lowered by noticing that many of the graph isomorphisms lead to inconsistent bijections. This is early noticed by Algorithm 2, allowing to discard several bijections. A more refined analysis will be performed in Section 4.3, where improvements to the Miller algorithm are proposed based on the observations here detailed.

Example 5. To better understand the isomorphism test based on cycle graphs, a run willl be performed in two STSs of samlle order. For this, consider two isomorphic triple systems: Since both systems are isomorphic, the cycle vector of both systems

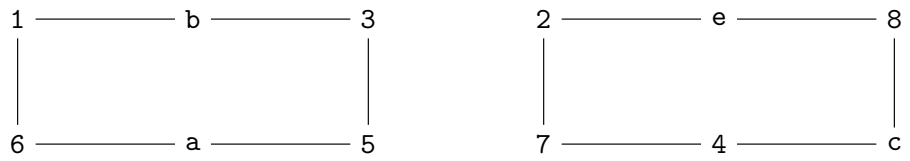
S_1: `000000011111122222333334445566667 88`
`123457c24578a4579b4579d59a9a789cbab`
`6da8b9e3cde9b68caeb68ce7deecaebdddc`

S_2: `000000011111122222333334445566778a`
`123458923589d3569b4567578abc79c8a9b`
`67bcaed4c7abea8decd98e69beedbaedccd`

is identical. In both cases, cycles of length 12 exist, and should be selected to provide a fast isomorphism search. However, to detail the use of the algorithms previously mentioned, the analysis will be carried out over the cycle graph G_{01} of S_1 and G_{09} of S', both consisting of two cycles of length 6. The graphs are shown in Figure 4.1.



(a) Cycle graph of S for the pair $\{0, 1\}$.



(b) Cycle graph of S' for the pair $\{0, 9\}$.

Figure 4.1: Cycle graphs for isomorphism technique.

We consider assigning the vertex sets in the order shown in Figures 4.1a and 4.1b. Additionally, `09d` $\in S_2$ must be the image of `016` $\in S_1$ under ϕ. We start considering

the partial bijection defined by assigning vertices in the following way

$$\begin{array}{llll} \texttt{0} \mapsto \texttt{0} & \texttt{6} \mapsto \texttt{d} & \texttt{3} \mapsto \texttt{b} & \texttt{b} \mapsto \texttt{5} \\ \texttt{1} \mapsto \texttt{9} & \texttt{2} \mapsto \texttt{1} & \texttt{a} \mapsto \texttt{3} & \texttt{5} \mapsto \texttt{a} \end{array}$$

Then, for every pair in the defined subset we find the third element looking in its containing block. For instance, consider the block `02d` of S_1. Since the image of the pair `02` is `01`, then the partial function ϕ is extending by setting $\phi(\texttt{d}) = \texttt{6}$. After one round the function is extended with

$$\begin{array}{lllll} \texttt{0} \mapsto \texttt{0} & \texttt{2} \mapsto \texttt{1} & \texttt{b} \mapsto \texttt{5} & \texttt{4} \mapsto \texttt{e} & \texttt{e} \mapsto \texttt{7} \\ \texttt{1} \mapsto \texttt{9} & \texttt{3} \mapsto \texttt{b} & \texttt{5} \mapsto \texttt{a} & \texttt{8} \mapsto \texttt{8} & \texttt{7} \mapsto \texttt{4} \\ \texttt{6} \mapsto \texttt{d} & \texttt{a} \mapsto \texttt{3} & \texttt{d} \mapsto \texttt{6} & \texttt{9} \mapsto \texttt{c} & \texttt{c} \mapsto \texttt{2} \end{array}$$

A fast inspection shows that in effect, this bijection is a suitable isomorphism of triple systems. The isomorphism between the remaining cycles is fixed by the isomorphism defined in the first components.

4.2.2 Joining cycle graphs

The first observation to take in account comes from whats has been developed in Section 3.2: the switching transformation can be used to create new instances with longer or shorter cycles. Having this in mind, the goal is to get suitable transformations that lowers the expensive step of finding an isomorphism between graphs with several cycles. This is achieved by selecting two isomorphic cycle graphs, say G_{xy} and $G'_{x'y'}$, from S and S' respectively, and transforming them into graphs H_{xy} and $H'_{x'y'}$ with few cycle graphs, ideally getting a single one. This allows to apply Algorithm 3 efficiently. A single step is required to pass from a cycle graph with several cycles to another with a single one. This step consists in performing suitable switching transformations to join the cycles into a single one.

Specifically, consider a system S and a cycle graph G_{xy} with connected components $C_1, \ldots, C_k$. For each component, pick a vertex $v_i \in C_i$. The system having a unique cycle is given by

$$T = sw^x_{v_1v_2} \circ \cdots \circ sw^x_{v_{k-1}v_k}(S)$$

The resulting STS T has indeed at least a cycle graph with a unique component. To verify this, define $T_0 = S$ and $T_i = sw_{v_iv_i+1}(T_{i-1})$ for $i = 1, \ldots, n-1$. The cycles $C_1, C_2 \subset G_{xy}$ are transformed into a unique cycle on the vertex set $V(C_1) \bigcup V(C_2)$. The remaining cycles are unchanged, as has already been observed in 3.2.2. Thus, the

cycle containing v_1 after i iterations is conformed by the vertex set $V = \bigcup_{j=1}^{i+1} V(C_j)$, having length $\sum_{j=1}^{i+1} |V(C_j)|$.

Finally, solving the isomorphism problem for the long cycles helps to solve the problem in the original systems. This idea is better understood in the commutative diagram shown in Figure 4.2. In this diagram we chose to denote by sw to the composition $= sw^x_{v_1v_2} \circ \cdots \circ sw^x_{v_{k-1}v_k}$ for simplicity.

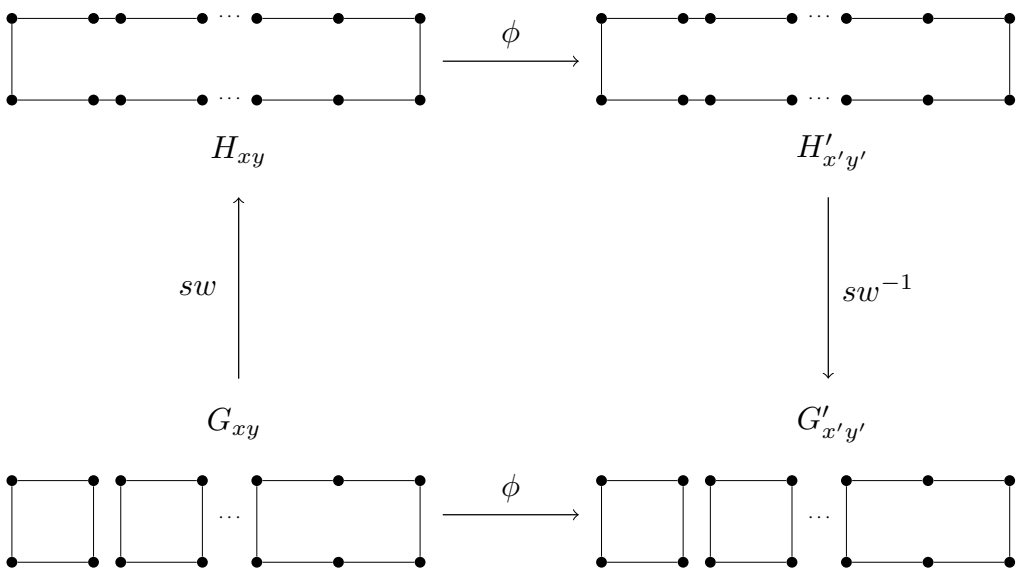


Figure 4.2: Solving isomorphism with cycle graphs.

In order to provide a correct algorithm, it should be observed that the switching transformation is not commutative, thus the ordering of the cycles C_i matters. The left side arrow of Figure 4.2 can be computed with an arbitrary order, however the right arrow should be selected carefully. In addition to the order, an additional difficulty lies in the process of selection of elements $\{x, y, v\}$ to perform $sw^v_{x,y}$, the resulting systems T and T' obtained from isomorphic systems S and S' must be isomorphic as well. To ensure that on each transformation, the resulting systems are again isomorphic, a closer inspection on how the cycles are transformed could be done to guarantee that at least, the cycle vectors are identical on every step. To this end, the result obtained in Proposition 6 can be beneficial. Nevertheless, the simplicity and fine results obtained by the improvements introduced in the next section moved us to employ the technique there proposed. Given this information, Algorithm 4 can be used to find an isomorphism between STSs provided that it exists.

Algorithm 4 Finding isomorphisms with cycle graphs

Require: Steiner triple systems S and S' of the same order.

Ensure: Isomorphism if exists or **false** otherwise.

1: Select $\{x, y\} \subset X$ at random.
2: Compute the cycle graph G_{xy}
3: Select $v_i \in C_i$ at random for each component $C_1, \ldots, C_k \in G_{xy}$
4: $T \leftarrow sw^{x}_{v_1 v_2} \circ \cdots \circ sw^{x}_{v_{k-1}, v_k}(S)$
5: **for** each $\{x', y'\} \in \binom{X}{2}$ **do**
6: **if** $G_{x',y'} \approx G_{xy}$ **then**
7: **for** each $\sigma \in \mathbf{S}_k$ **do**
8: Select $v'_i \in C'_{\sigma(i)}$ at random
9: $T' \leftarrow sw^{x}_{v'_1 v'_2} \circ \cdots \circ sw^{x'}_{v'_{k-1}, v'_k}(S')$
10: $\phi \leftarrow$ TestIsomorphism(T,T') *#use Alg.3 on G_{xy} and $G_{x'y'}$*
11: **if** ϕ is an isomorphism **then**
12: **return** ϕ
13: return **false**

4.2.2.1 Complexity

Let us recall that the notation $sw^{v}_{xy}(S)$ is only another way to express $sw_{xy}(S, C)$, where $v \in V(C)$. Then the selection of the vertex sequence v_i aforementioned can be arbitrary within each corresponding cycle C_i. As to the complexity of the algorithm, the relevant process starts corresponds to iterations in steps 5 and 7. Step 5 iterates over every possible pair, totaling $\frac{n(n-1)}{2}$ at most. The second iteration is necessary to cover all possible transformations, following the observation on the non-commutativity of the switching operation. Finally, test isomorphism is linear on n, since the test is made on one cycle. The maximum number of iteration is then $k! n \frac{n(n-1)}{2}$.

Comparable to the cycle graph algorithm presented in section 3, the expensive step lies in the amount of possible permutations in the component set of the cycle graph G_{xy}.

4.3 Improving Miller's algorithm

An significant improvement of Miller's algorithm has been presented in [8]. The technique employed consists in computing a representation for a STS, known as the *canonical form*, which is unique up to isomorphism, i.e., two non-isomorphic STSs will have

a different canonical form. To obtain a canonical representation, a lexicographic ordering of the triples is considered to define an ordering in the set of all STS(n). Since a STS(n) has roughly $n!$ different representations, obtaining the canonical form requires comparing all possible isomorphisms, which requires an exponential amount of comparisons. This upper bound is lowered with the help of Miller's algorithm, thus requiring only the permutation of generators. The final complexity of the algorithm reduces to $O((\log n)^2)$ running time while requiring $O(n^{\log n+2})$ processors to achieve this bound.

This is a considerable improvement, and considering nowadays technology, this poses a huge disadvantage for using the isomophism problem on STSs for cryptographic purposes. In addition to this, it is conjectured that the average-case complexity of Miller's algorithm is polynomial, something that has been already pointed out in Section 4.2. Notwithstanding, a detailed analysis is required in order to find instances where the number of processors required makes the task infeasible. In this section, improvements to the (sequential) Miller's algorithm are proposed to find suitable instances. These improvements are guided by a more detailed analysis of both the cycle graphs and their relation to the Steiner quasigroups generators. Results previously obtained concerning cycle graphs will be utterly useful for this purpose.

Cycle graphs are useful to guide the search of the generators of the associated quasigroup. For instance, take the isomorphic systems S_1 and S_2 given in Example 5, and the cycles shown in Figure 4.1. It is an easy task to verify that the elements $\{\texttt{2,3,a}\}$ are generators for Q_{S_1}. These elements are mapped into the set of generators $\{\texttt{0,9,1}\}$ (in this order) of Q_{S_2} by the isomorphism found. both triples are consecutive edges of $G_{\texttt{0,1}}$ and $G_{\texttt{0,9}}$ respectively. On the other hand, the triple $\{\texttt{0,1,2}\}$ is also a set of generators of Q_{S_2}, but there is no bijection of X sending $\{\texttt{2,3,a}\}$ into $\{\texttt{0,1,2}\}$ that yields an isomorphism ϕ already found in Example 5. This can be easily verified by hand, but also because the later triple does not appear as consecutive vertices of any isomorphic cycle graph of S_2.

The current approach is based on a more intelligent selection of the generators of the quasigroup associated to STSs. As it has been mentioned in Section 4.2, isomorphism tests on systems with big cycle graph components can be executed efficiently. This fact will be used to provide a substantial improvement for the Miller's algorithm.

Proposition 11. *Let $S = (X, \mathcal{B})$ be a STS and Q_S its Steiner quasigroup. Consider a pair of elements $\{x, y\} \subset X$ and the cycle graph G_{xy}. Let $C \subset G_{xy}$ be a connected component with edges $\{v_0, v_1\}$ and $\{v_1, v_2\}, v_0 \neq v_2$. Then*

$$\langle V(C) \rangle = \langle v_0, v_1, v_2 \rangle.$$

Proof. Observe that $x, y \in \langle v_0, v_1, v_2 \rangle$ (directly from the definition of cycle graph). The complete cycle is obtained by observing that

$$v_{i+1} = \begin{cases} v_i x \text{ if } y = v_{i-1}v_i \\ v_i y \text{ if } x = v_{i-1}v_i \end{cases}$$

for $i = 3, \ldots, |C| - 1$. In fact, for any element in $v \in V(C)$, it can be seen that $\langle x, y, v \rangle = \langle v_0, v_1, v_2 \rangle$. □

Additionally, the Steiner subsystem T generated by the triples $\mathcal{T}_{xy}^{x}$ involved in the cycle component C is related to the subquasigroup generated by $Q_T = \langle v_0, v_1, v_2 \rangle$. This is, the bocks in $\mathcal{T}_{xy}^{v}$ generate essentialy the same object than the vertices v_i. Any 3 vertices of a cycle of length $v - 3$ generate the full Steiner quasigroup, i.e., it has only 3 generators. Considering this observation, testing isomorphism on two random STSs is generally performed efficiently, since the number of generators in the Miller's test is extremely low. Only 3 in a vast amount of the cases. Avoiding these cases will be the main objective for the next section.

The algorithm here proposed makes use of the aforementioned result by using bigger cycles first. The first step is to get a small set of generators. Algorithm 5 gets generators considering a cycle graph with few components as possible. To find a generator set with minimum size, we should proceed by performing Algorithm 5 for every pair $\{x, y\} \subset X$, however, by considering a cycle graph as determined in step 1, it is guaranteed that the size is close to the minimum.

Algorithm 5 Getting a generator set

Require: Steiner triple system S and a pair $\{x, y\}$.

Ensure: A set of generators of the Steiner quasigroup of S.

1: $C_1, \ldots, C_k$ connected components of G_{xy} with $|C_i| \geq |C_{i+1}|$.
2: $V, \leftarrow \{v_1, \ldots, v_k\}$ with v_i selected at random from C_i
3: $R \leftarrow \{x, y\}$
4: $Q \leftarrow \langle x, y \rangle$
5: **while** $V \neq \emptyset$ **do**
6: $\quad i \leftarrow$ minimum sub-index of $v_i \in V$
7: $\quad R \leftarrow R \bigcup \{v_i\}$
8: $\quad Q \leftarrow \langle R \rangle$
9: $\quad V \leftarrow V - Q$
10: **return** R

The algorithm is based on the Miller's algorithm, but in this case we start by adding the elements that generate bigger sub-quasigroups first. Though the maximum number of elements in V is $\left\lfloor \frac{n-3}{4} \right\rfloor$, the maximum number of elements in the generator set is $\log n$.

The isomorphism search algorithm starts at finding a set of generators from one of the Steiner quasigroup. Then, from isomorphic cycle graphs generators are also extracted and bijections between generator sets are verified to check if an isomorphism is obtained. Suitable bijections considered in step 6 of Algorithm 6 must consider the length of cycle C_i which contains v_i. This is, if $|C_i| \neq |C'_j|$ then $\phi(v_i) \neq v_j$.

Algorithm 6 Improved version of Miller's algorithm

Require: Steiner triple systems S and S' of the same order.

Ensure: Isomorphism if exists or **false** otherwise.

1: Let $\{x, y\}$ be such that G_{xy} has minimal number of connected components
2: $R \leftarrow$GENERATORS(S, x, y) *#use Algorithm 5*
3: **for** each $\{x', y'\} \subset X'$ **do**
4: **if** the cycle lists of G_{xy} and $G'_{x'y'}$ are identical **then**
5: $R' \leftarrow$GENERATORS(S', x', y') *#use Algorithm 5*
6: **for** each suitable bijection $\phi : R \to R'$ **do**
7: **if** ϕ defines an isomorphism **then**
8: **return** ϕ
9: return **false**

The main difference of Algorithm 6 with Miller's algorithm consists in testing a set of generators of S' related to the fixed set of generators of S related by their membership to cycles of the same length. This is done in step 6, where, at most, $(\log n)!$ different bijections should be tested. The corresponding running time does not differ from that of Miller's technique. As we have already noted, for the average case, the number of generators is extremely low, and the original algorithm could outperform the improved one, mainly because the latter requires computing a series of cycle graphs to find the best option. To reduce this computational burden, only a small sample of the possible $\frac{n(n-1)}{2}$ pairs is selected and the best graph for the test is kept. Practical examples will be performed and analyzed in Chapter 5 to provide further evidence on the choice of suitable systems for the ZKP protocol to be proposed.

Thus, in addition to requiring small cycles in every cycle graph, it is also desirable that an important number of cycle graphs G_{xy} are isomorphic. Otherwise, the

smallest number of isomorphism classes in the set of cycle graphs can be used to find the isomorphism between systems. This requirement is an important one since Algorithm 6 is designed to discard unsuitable generators (step 4), improving the efficiency for this instances substantially.

Chapter 5

Building difficult instances

In the previous chapter, desirable characteristics for Steiner triple systems to provide difficult isomorphism instances were established. The characterization emerged as the result of examining the best general algorithms that solve the STS isomorphism problem. A summary of the characteristics is listed below:

- The Steiner quasigroup of the STS must have as many generators as possible.
- Cycle graphs must have several connected components, and ideally, every connected component should have the same length.
- The isomorphism test given in Alg. 6, discards non-isomorphic graphs. Thus, every cycle graph should be isomorphic, i.e., the STSs should be uniform.

Points 2 and 3 can be relaxed to analyze a broader group of instances. For point 2, we consider systems that allow few different lengths of its cycles. This is, for every cycle graph G_{xy} with cycle list $(c_1, \ldots, c_l)$ we have $\#\{c_i \neq 0\} < \alpha$ for a small α. Point 3 can be relaxed to consider "almost" uniform systems. Such systems will be regarded as those where the cycle vector has few non-zero entries. Techniques to build instances with these characteristics will be addressed in 5.2.

The first class of instances considered are those containing only cycles of length 4, the ideal case for the instances we are looking for. This case has been briefly addressed in 3.2, and as mentioned, unfortunately, there is a unique instance for each admissible n (up to isomorphism), which happens to be isomorphic to a projective space $\mathrm{PG}(2, k)$. For this reason, this class of systems are not suitable for cryptographic purposes. Any base of the system $\mathbb{F}_2^k$ can be used to construct the required isomorphism, which reduces the complexity drastically, even by applying the Miller's algorithm, since

any bijection between two sets of generators of the Steiner quasigroup defines an isomorphism.

The next set of systems to be considered are those having every cycle of length 6. In this case, a family of systems, known as Hall triple systems (HTS), are well suited for this task.

5.1 Hall triple systems

Among the instances that pose difficulties for the isomorphism tests previously proposed, some of the most promising due to its characteristics (uniformity, number and size of cycles) are the Hall triple systems.

Definition 11. A *Hall triple system* is a STS where any two intersecting blocks generate a subsystem isomorphic to STS(9).

Then, any three elements of the related Steiner quasigroup will generate a subquasigroup of size 3 (if every element lies within the same block) or 9. In [27] Hall proves that every HTS has order 3^k for $k > 1$. It is worth mentioning that systems of this kind are rare, but the exact number or an approximate density is unknown. Up to now, all that is known is summarized in the following table, extracted from [9]).

Order:	3^2	3^3	3^4	3^5	3^6	3^7	3^8
Number of HTS:	1	1	2	2	4	13	≥ 45

The table shows the number of existing Hall triple systems of size 3^k for $k < 9$. For higher dimensions, the results are currently unknown.

Regarding the number of generators of such systems, Beneteau shows that any two minimal generator sets of a HTS have the same cardinality [3]. A HTS is said to have *dimension* m if the cardinality of any minimal generator set is $m + 1$. Unfortunately, the dimension of an HTS is not always close to $\log n$, as desired. In fact, the maximum number of generators for a HTS of order $n = 3^k$ is $k + 1$, and it holds only for one system of order 3^k (up to isomorphism). Table 5.1 (extracted from [9]) summarizes the number of non-isomorphic HTSs of dimension d for small d. It is possible to see that instances of dimension close to $k + 1$ are numerous, and will be considered for the proof system specification.

Dimension \ Order	3^4	3^5	3^6	3^7	3^8
3	1	0	0	0	0
4	1	1	1	1	4
5	0	1	2	6	≥ 17
6	0	0	1	5	≥ 13
7	0	0	0	1	11
8	0	0	0	0	1

Table 5.1: Dimension vs order.

The second characteristic we need to verify is the length of the components of cycle graphs. Fortunately it is not difficult to verify that the cycle graphs in these type of systems show good behaviour.

Proposition 12. *Every cycle graph of a HTS is a disjoint union of 6-cycles.*

Proof. Consider an HTS $S = (X, \mathcal{B})$ and a subset $\{x, y, v\} \notin \mathcal{B}$. By definition, these 3 points generate a subsystem of size 9 and contains the cycle G^v_{xy}, therefore has length 6. Since any cycle is uniquely determined by 3 non-collinear elements, the result follows. □

Once we have identified HTSs as a reliable source of difficult instances, it is required to construct systems of this nature to provide practical examples. To this end, we require of further characterizations that allow to implement better suited construction techniques. The following sections address this problem.

5.1.1 Hill-climbing algorithm for HTS

Hill-climbing algorithms are iterative procedures applied frequently to find approximate solutions of optimization problems, such as the well-known traveling salesman problem [34]. In the field of combinatorial designs, a fast implementation to construct STSs was provided by Stinson [53], and several other have been designed to build systems with specific characteristics, such as ortogonality [20], Pasch-free [25] or directed triple systems [33].

The original hill-climbing algorithm designed by Stinson defines the *live points* as elements $x \in X$ that have not occurred in $\frac{n-1}{2}$ blocks. If the size of a partial STS is less than $\frac{n(n-1)}{6}$, then live points can be found. On each iteration, a new live point is found and a new block containing it is created. The total size of the block set

increases or remains, but do not decrease. The procedure proposed by Stinson for STS construction is detailed in Algorithm 7.

Algorithm 7 Hill-climbing algorithm for construction of Steiner triple systems

Require: Admissible integer n.

Ensure: A STS(n).

1: $\mathcal{B} \leftarrow \emptyset$
2: **while** $|\mathcal{B}| < \frac{n(n-1)}{6}$ **do**
3: Pick a random live point x
4: Pick random y, z which have not occurred with x
5: $B_0 \leftarrow \{x, y, z\}$
6: **if** y, z have not occurred in a block of $\mathcal{B}$ **then**
7: $\mathcal{B} \leftarrow \mathcal{B} \bigcup \{B_0\}$
8: **else**
9: $B_1 \leftarrow$ block containing $\{y, z\}$
10: $\mathcal{B} \leftarrow \mathcal{B} \bigcup \{B_0\} - \{B_1\}$
11: **return** $(\{1, \ldots, n\}, \mathcal{B})$

Additional restrictions in the hill-climbing construction are included to guarantee that the output STS is in fact a HTS. A useful result in this direction can be found in [46]. Petelczyc shows that a STS is a HTS if and only if it does not contain neither C_{16}-configurations (Pasch) nor C_A-configurations, the later defined as any $(8, 5)$-configuration isomorphic to $\{012, 034, 056, 137, 246\}$. This observations and knowing that every cycle graph must contain only cycles of length 6 allow us to define a hill-climbing algorithm for the construction of such systems.

Similar to Stinson's algorithm, at every iteration a new block is considered as candidate to be added to the partial STS, but two tests must be performed to avoid that Pasch or C_A-configurations are created with the newly created block. Additionally, instead of starting from scratch, the algorithm can be initialized with a partial system $\mathcal{B}$, which should be a suitable HTS-subsystem. The input subsystem must not be modified. Algorithm 8 shows the steps to be performed to get a HTS.

Unfortunately, at every iteration there is a chance of leaving the partial STS unmodified, which arises after selecting a pair x, y already contained in a block B_1 and choosing a new block B_0 that introduces any of the undesirable configurations. A upper limit in the number of iterations could be added to avoid an infinite loop.

A more detailed inspection of the verification performed in step 13 shows that this process might be a source of computational burden. A straightforward verification

consists in selecting every subset $\mathcal{S}$ of size 3 of $\mathcal{P}$ and verify if this is isomorphic to the quadrilateral. One way to prove this is by verifying that:

1. the union of the four blocks is a set of size 6,

2. every element appears exactly twice.

However, since the Pasch configuration is the unique $(6, 4)$-configuration that can appear in a STS, only the first condition must be verified. Moreover, observe that two disjoint blocks cannot be part of a quadrilateral simultaneously. Thus only blocks whose intersection with B_0 is not empty must be considered.

Algorithm 8 Hill-climbing algorithm for construction of Hall triple systems

Require: Exponent k and partial HTS $\mathcal{B}$.
Ensure: A HTS of order 3^k or **false** if construction fails.

1: $\mathcal{P} \leftarrow \mathcal{B}$
2: $n \leftarrow 3^k$
3: **while** $|\mathcal{P}| < \frac{n(n-1)}{2}$ **do**
4: Pick a random live point x
5: Pick random y, z which have not occurred with x
6: $B_0 \leftarrow \{x, y, z\}$
7: $B_1 \leftarrow \emptyset$
8: **if** y, z have occurred in a block of $\mathcal{B}$ **then**
9: $B_1 \leftarrow$ block containing $\{y, z\}$
10: **if** $B_1 \in \mathcal{B}$ **then**
11: **goto** 4
12: $\mathcal{P} \leftarrow \mathcal{P} - \{B_1\}$
13: **if** $\mathcal{P} \bigcup \{B_0\}$ is free of Pasch and C_A configurations **then**
14: $\mathcal{P} \leftarrow \mathcal{P} \bigcup \{B_0\}$
15: **else if** $B_1 \neq \emptyset$ **then**
16: $\mathcal{P} \leftarrow \mathcal{P} \bigcup \{B_1\}$
17: **return** $(\{1, \ldots, n\}, \mathcal{P})$

The verification for C_A-configurations will require more steps. To evaluate the new block, we start by choosing every $\mathcal{S} \subset \mathcal{P}$ of size 4 such that $\mathcal{S} \bigcup \{\{x, y, z\}\}$ is a $(8, 5)$-configuration. Then the size of the union of the blocks must be 8. Two different invariants are considered:

- the *degree* of the elements in the partial STS $S \bigcup \{\{x, y, z\}\}$. This is, the number of blocks containing an element. For a C_A-configuration, the sorte list of degrees must be equal to (3,2,2,2,2,2,1,1)

- Considering the graph $G = (V, E)$ where V is the set block and

$$E = \left\{ (B_1, B_2) \mid B_1, B_2 \in \mathcal{B}, B_1 \bigcap B_2 = \emptyset \right\},$$

  the sorted list of degrees in G must be (4,4,3,3,2).

These verifications turn out to be sufficient to verify that a set fo 5 blocks is not a C_A-configuration.

A second point of interest is the initial subsystem considered in the algorithm. Instead of starting from an empty set as the original hill-climbing algorithm, it is possible to provide a good starter by using, for example, a HTS of a big size or, knowing that any two intersecting blocks in a HTS generate a STS(9), by providing several subsystems of size 9 as base building blocks. In the former case, a HTS of size 3^k can be input to Algorithm 5.1 to obtain one of a bigger size. In addition to the size, the number of generators will be bigger. For the later approach, 3^{k-2} disjoint STS(9) can be created by taking subsets of 9 disjoint elements.

Unfortunately, due to the size of a STS($3k$), even for moderate values of k, and the scarcity of such systems, the algorithm might require a long time before finding new instances. For this reason, we study additional direct construction techniques. The main disadvantage with the direct approaches is that only a subset of the existing HTSs can be constructed, which might limit the number of instances that can be considered.

5.1.2 Algebraic construction of HTS

The most basic of the algebraic constructions for HTS consists in considering the set of lines of a vector space $\mathbb{F}_3^k$ as the block set. A line consists of 3 elements (x, y, z) such that $x + y + z = 0$, where 0 must be regarded as the vector with zero entries. In the same fashion as the analysis done for $PG(2, k)$, any set of $k + 1$ generators of the associated quasigroup can be used to find an isomorphism between isomorphic systems, which makes the isomorphism problem easily solvable and not suitable for cryptographic purposes. This system is called the *affine Hall triple system*, denoted AG$(3, k)$. The affine HTS surges as a special case of the following construction [9, 28.5].

Consider the vector space $\mathbb{F}_3^{k+1}$ with $k \geq 3$ and choose a basis $e_0, \ldots, e_k$. Let $\alpha = (\alpha_0, \ldots, \alpha_k)$ and $\beta = (\beta_0, \ldots, \beta_k)$ be such that

$$x = \sum_{i=0}^{k} \alpha_i e_i; \;\; y = \sum_{i=0}^{k} \beta_i e_i.$$

Then z is defined as the element such that

$$x + y + z = \sum_{i<j<k} \Lambda_{ijk}(\alpha_i - \beta_i)(\alpha_j\beta_k - \alpha_k\beta_j)e_0 \tag{5.1}$$

where $(\Lambda_{ijk})_{i<j<k}$ is a sequence of elements of $\mathbb{F}_3$. By setting $\Lambda_{ijk} = 0$ for every $i < j < k$, the right part of (5.1) is always zero. Thus, coincides with the definition of the affine HTS.

Example 6. Considering the above construction, let $k = 4$ and

$$\Lambda_{ijk} = \begin{cases} 1 & \text{if } i = 1, j = 2, k = 3 \\ 0 & \text{otherwise.} \end{cases}$$

Then the STS(3^4) obtained is an HTS that is not isomorphic to AG$(3, 4)$. In fact the dimension of AG$(3, 4)$ is 5 whereas the HTS constructed is 4.

The characteristic that makes the isomorphic problem in AG(3,k) efficiently solvable is that a bijection between any pair of sets of generators of two isomorphic systems can be used to build an isomorphism of systems. This is not the case for non-affine systems. Thus, algorithms require in general more time to find suitable generators. Experimental support of this assertion will be offered in Section 5.3.

5.2 Non-uniform instances

Every system considered so far is uniform: all cycle graphs are pairwise isomorphic. In the case of projective geometries, every cycle graph is the union of cycles of length 4. For Hall triple systems, the length is 6 for every cycle.

In Section 3.2.1, an approach to join and split cycles in cycle graphs using the switching transform was described. Starting with an STS S with cycles of small length, it is possible to provide a new instance with cycles of size similar to those obtained in the former system. Take for instance a projective geometry $S =$PG(2^k). Consider any pair of different elements x, y and apply the cycle switch $s^v_{x,y}$ for a suitable v. The resulting system $S' = sw^v_{xy}(S)$ will contain some cycles of length 8.

For instance, consider $k = 5$. The system PG$(2, 5)$ consists of 31 elements and each cycle graph of $\frac{31-3}{4} = 7$ connected components. The unique non-zero element of the cycle vector corresponds to the tuple $(4, 4, 4, 4, 4, 4, 4)$, and its value is $\frac{31 \cdot 30}{2} = 465$. The system S_2 which results after applying sw_{xy}^{v} will have 144 graphs with cycle list $(4, 4, 4, 4, 4, 8)$. By symmetry, any suitable triple (x, y, v) will output an isomorphic result.

Though the change is small and the number of generators is reduced with respect to the original STS(2^k), experimental results detailed in 5.3 show that these systems provide difficult instances for the Miller algorithm, and in some cases, for the improved algorithms developed in Section 4.3.

Another useful tool for providing systems with small cycles relies on the transformation performed by using trades. We have seen in Section 3.4 that we can obtain new systems by exchanging suitable isomorphic subsets from a pair of STSs of the same order. For $S =$PG$(2, k)$ it is easy to proceed: consider a subsystem T of order 7 of S and obtain a random permutation $T' = \phi(T)$. The new system is given by $S' = (S - T) \bigcup T'$. The biggest cycle gathered in this way is 12, while must of the cycles remain with size 4.

In a similar fashion, Hall triple system can lead to new systems, where the cycles and the number of generators do not change drastically, either by applying a random switching transformation or by using trades. Since any two intersecting blocks generate a subsystem of size 9, two random blocks can be chosen to get a STS(9), which is randomly permuted to replace the original subsystem.

With all the new instances and the proven resistance to isomorphism tests, it is possible to consider the obtained systems for proposing cryptographic protocols. Nonetheless, the improvements achieved for the original Miller's algorithm show that the complexity of many instances can be drastically lowered, which would require much bigger systems, something undesirable for practical scenarios.

5.3 Experimental results

In this section some experimental results of instances addressed with both, the original Miller's algorithm and the improved algorithm, are presented. Table 5.2 shows the available resources of the equipment used to perform the tests described later.

In the first set of tests, systems derived from PG(2,k) are considered. For each k, two different types instances are constructed and analyzed:

- PG$(2, k)$, whose minimal generating sets has k generators and every cycle is

Resources	Characteristics
OS	Ubuntu 18.04
RAM memory	4 GB
Processor	Intel Core 2 Quad CPU Q8200 @ 2.33GHz x 4
Language	Python v3.5

Table 5.2: Computational resources.

isomorphisc to a union of disjoint 4-cycles,

- a system derived from $\mathrm{PG}(2, k)$ by applying a cycle switch operation at random. We have been able to find minimal generating sets of size $k - 1$. Some cycle graphs contain exactly one cycle of length 8.

For both cases, around of 20 examples are runned to obtain an average of the running time for both, the projective and the transformed instances. An observation worth mentioning is that running times can differ a lot even for instances of the same size. This can be due possibly to the different existing isomorphisms between systems, but also on the initial selection of the first generating set, a phenomenon that requires of further research.

It has been previously stated that $\mathrm{PG}(2, k)$ is not a difficult instance, even for the original Miller's algorithm, mainly because any set of generators can be paired to produce a valid isomorphism. However, in the second case it does not happen, as it can be seen from the results of the algorithm executions. For the first class of instances, it is possible to see that the performance is comparable in both cases. Even more, the original algorithm outperforms the modified version here developed. A very different behavior is observed for the transformed system. The running time varies from just a few seconds up to several minutes for small instances for the improved algorithm, while the running time for the original Miller is extremely high, as shown in Table 5.3. An horizontal line means that the algorithm has not been able to provide a result within a day. The executions make clear that even a small modification by means of the cycle switch, turns an easy instance into difficult ones for Miller's algorithm. The main advantage in our proposal lies in the fact that the search of possible bijections is made from possible generating sets, while Miller's is made from every possible set of elements. As the order n grows, the amount of possible combinations grows.

The second set of instances are created from the algebraic HTS construction studied in 5.1.2, and in the same fashion as the first set of tests, 20 executions per order are

		Running time (seconds)	
		Miller Algorithm	Improved Miller
$k = 4$	PG$(2, k)$	0.0004	0.0037
	Switched	0.0407	0.0022
$k = 5$	PG$(2, k)$	0.0020	0.0395
	Switched	2.692	0.734
$k = 6$	PG$(2, k)$	0.0191	0.029
	Switched	–	32.771
$k = 7$	PG$(2, k)$	0.044	0.083
	Switched	–	–

Table 5.3: Results of executions of isomorphism tests.

performed to obtain the average running time. It is worth noticing that the behavior of both algorithms is similar to what has been shown before: Miller's algorithm seems to be better suited for the affine case AG$(3, k)$, while the improved algorithm shows a huge advantage as soon as k starts growing, as seen on Table 5.4. The exponential increment of the running time in the affine case is related to the number of operations needed to generate the quasigroup. It is possble to see the running time in the non-affine case increases even more.

		Running time (seconds)	
		Miller Algorithm	Improved Miller
$k = 4$	Affine	0.018	2.407
	Non-affine	0.0253	0.0236
$k = 5$	Affine	0.222	0.279
	Non-affine	0.523	0.582
$k = 6$	Affine	3.702	3.565
	Non-affine	–	10.149
$k = 7$	Affine	30.035	36.023
	Non-affine	–	694.9
$k = 8$	Affine	424.9188	–
	Non-affine	–	–

Table 5.4: Results of executions of isomorphism tests.

The results obtained from both series of experiments show that the characterization carried out in Chapter 5 is correct. STSs whose cycle graphs consist of several connected components will provide difficult isomorphism instances. This difficulty has been measured by applying the best algorithm known to address this problem. An alternative improvement of such algorithm has worked better for the instances obtained from a cycle switching construction. From this result, the need of a more detailed analysis is clear, and as we have already stated, a possibility lies in the study of the generating sets, where a more intelligent selection might be better suited for isomorphism testing. For example, we know that an element belonging to a cycle of length l must be paired with a similar element. This observation has been used in the improved Algorithm 6. A more refined selection might be done by counting the number of cycles of length l that contain this same element, which can be used as an invariant, thus reducing the possible bijections by performing a closer inspection to the cycle graph information. This ideas will be considered for future research.

Chapter 6

Zero Knowledge Proof Systems Based on STSs

One of the first and most known zero-knowledge proof is based on the isomorphism problem in graphs (GI) [2]. The main difficulty in this approach resides in the fact that the GI problem can be easily solvable for the average case with state-of-the-art solvers such as nauty or Traces [38], saucy [5], or bliss [29] among others, thus purely random generators are not suitable for cryptographic purposes. In addition to these results, Babai [1] has proposed a novel technique reducing the complexity of GI to quasi-polynomial time (with a running time of $2^{O((\log n)^c)}$). Notwithstanding, the general problem is still difficult, and, as noted in [43], some of the hardest instances arise from combinatorial objects such as strongly regular graphs, block designs, and coherent configurations.

We have already studied how to create hard instances for the isomorphism problem in Steiner triple systems. We are ready to propose some ZKP systems based on this and other difficult problems.

Problem 1: consider two different Steiner triple systems S_0, S_1 (not necessarily isomorphic). Find a sequence of triples $[\{x_i, y_i, v_i\}]_{i=1}^{m}$ such that

$$S_1 = sw_{x_m,y_m}^{v_m} \circ \cdots \circ sw_{x_1y_1}^{v_1}(S_0)$$

Another related problem can be formulated as follows.

Problem 2: consider two different Steiner triple systems S_0, S_1 (not necessarily isomorphic). Find a bijection $\phi : X \to X$ and a triple $\{x, y, v\}$ such that

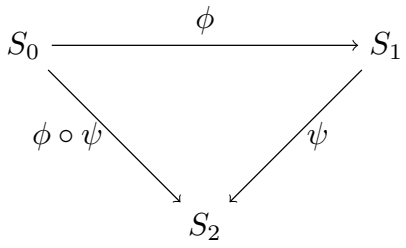
$$S_1 = \left[\phi \circ sw_{xy}^{v}\right]^m (S_0).$$

Regarding problem 1, it is conjectured that a composition of switching transformations connects any two isomorphism classes. Considering the vast amount of isomorphism classes, addressing this problem by brute force is unfeasible. It is unknown whether a similar conjecture can be done for the second case. Note also trades can be taken into consideration since switchings can be regarded as a particular case of trades. However, the difficulty of encoding properly such transformations make it preferable to work with switchings, since only three numbers are required to express the specify the transformation.

In the following sections, some zero-knowledge proofs are introduced for STSs. Sections 6.1 and 6.2 are based on analogous descriptions for graph isomorphism instances. In section 6.3, a novel proof system is introduced, which makes use of the cycle switching transformation. For this reason, a more detailed analysis of this procedure is provided. Finally, we address the implementation concerns that must be tackled to provide an implementation of the proof systems to be presented.

6.1 A ZKP based on isomorphism problem

A straightforward approach to defining a proof system is based on the isomorphism problem. Isomorphisms can be defined for several mathematical objects, and in those where the problem of isomorphism is difficult, zero-knowledge proof systems can be defined, based mainly on the following commutative diagram.



The security of the protocol relies on the hardness of computing $\phi \circ \psi$ without knowing ψ, even knowing the isomorphism ϕ. In fact, the difficulty is comparable to computing the isomorphism ϕ.

For this purpose, well-known problems, such as the Graph Isomorphism problem (GI) [2] and, recently, the Isomorphism Problem for Multivariate Polynomials (IP) [44]. Difficult instances for Steiner triple systems, already discussed in Chapter 5, can be considered to provide a similar protocol. This protocol, in an analogous fashion to that provided for GI, starts by selecting a suitable system $S_0 = (X, \mathcal{B})$

and a random permutation ϕ. The pair $S_0, S_1 = \phi(S_0)$ is published, while the isomorphism ϕ is kept secret. The interaction between prover and verifier is described next.

6.1.0.1 Key generation

Choose a STS $S_0 = (X, \mathcal{B}_0)$ and a random permutation $\phi : X \to X$. The pair (S_0, S_1) where $S_1 = \phi(S_0)$ is the public key. The private key consists of the triple (S_0, S_1, ϕ_1).

6.1.0.2 Authentication process

1. *Prover.* Select a random permutation ψ. Compute $S_2 = \phi(S_1)$. Send S_2 to V.
2. *Verifier.* Select a random bit $\alpha \in \{0, 1\}$ and send it to P.
3. *Prover.* Let $\psi_0 = \psi \circ \phi, \psi_1 = \psi$. Send $\psi' = \psi_\alpha$ to V according to the received α.
4. *Verifier.* If $\psi'(S_\alpha) = S_2$ approve, otherwise reject.

For the protocol to be secure, hard isomorphism instances, as studied in 5, must be considered.

6.2 A ZKP based on non-isomorphism problem

The existence of a graph non-isomorphism protocol [22] allows to define a similar protocol for STSs. Both, prover and verifier decide a pair of non-isomorphic triple STSs S_0, S_1 to interact.

6.2.0.1 Key generation

Choose two random non-isomorphic STSs S_0, S_1 as public key. The private key is a way to decide efficiently $S_0 \ncong S_1$.

1. *Verifier.* Choose a random bit $\alpha \in_R \{0, 1\}$ and a random bijection $\phi : X \to X$. Constructs $T = \phi(S_\alpha)$ and sends to the prover.
2. *Verifier* In addition to T, the verifier must perform a zero-knowledge proof to show that he knows which of S_0, S_1 is isomorphic to T.

3. *Prover.* Performs an isomorphism test to decide $\beta \in \{0, 1\}$ such that $T \cong G_\beta$. Send β to the verifier.

4. *Verifier* Verifies that, in effect, $\alpha = \beta$.

In this case, non-isomorphic systems where invariants do not provide a way to differentiate them are desirable. This is the case of Hall triple systems with the same number of generators, which additionally have all cycle graphs isomorphic.

6.3 A ZKP based on the switching transformation

The aim of this section is to introduce a novel proof-system built adopting a combinatorial approach. To define a one-way function, we consider the difficulty of two problems already detailed at the beginning of the chapter: finding an isomorphism and inverting cycle switchings. To understand the challenge step of the authentication protocol, the commutative diagram shown in Figure 6.1 might be of help. The basis of the challenge resides in the fact that, $sw^{v}_{xy}(S) \cong sw^{v'}_{x'y'}(T)$ if an isomorphism $\phi : S \to T$ exists with $x' = \phi(x), y' = \phi(y), v' = \phi(v)$.

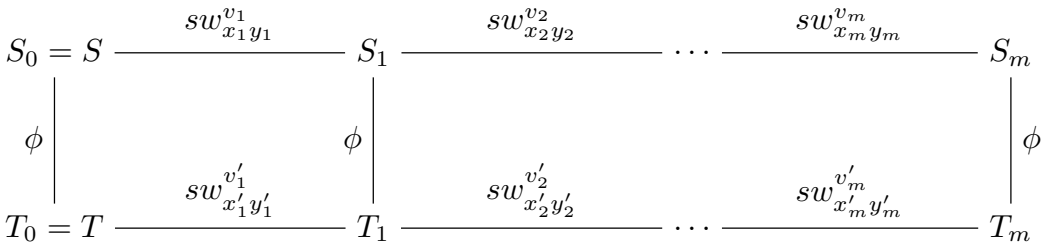


Figure 6.1: Cycle switching in isomorphic graphs.

After each round, the prover will answer correctly to the challenge by providing the isomorpism ϕ or the series of transformed triples that define cycle switch elements $[\{x'_i, y'_i, v'_i)]_{i=1}^{m}$ needed to transform T_0 into T_m with switching cycles. The procedure is explained next.

6.3.1 Authentication protocol

6.3.1.1 Key generation

To generate the key pair, the following steps are performed:

1. A Steiner triple system $S = (X, \mathcal{B})$ of order n is chosen at random (from a set of difficult isomorphism instances) and a list of triples $L = [\{x_i, y_i, v_i\}]_{i=1}^{m}$ such that $\{x_i, y_i, z_i\} \in \binom{X}{3} - \mathcal{B}$.

2. Compute the sequence $[S_i]_{i=0}^{m-1}$ of STS(n) such that

$$S_i = \begin{cases} S_0 & \text{if } i = 0 \\ sw_{x_i y_i}^{v_i}(S_{i-1}) & \text{if } i > 0. \end{cases}$$

The public key consists of the pair (S_0, S_m), while the private key consists of (S_0, S_m, L).

6.3.1.2 Authentication process

The verifier shall be convinced that the prover possesses the private key. The following interaction performed to this end.

1. *Prover*: selects randomly a permutation $\phi : X \to X$ and builds the STS(n) T_0, T_m isomorphic, respectively, to S_0 and S_m. Sends (T_0, T_m) to the verifier.

2. *Verifier*: chooses randomly a bit $\alpha \in \{0, 1\}$ and sends it to the prover as challenge.

3. *Prover.* The prover gets α and responses accordingly:

   $\alpha = 0$**:** the isomorphism $\phi : S_0 \to T_0$

   $\alpha = 1$**:** the list L' of triples to transform T_0 into T_m. This is computed as $L' = [\{\phi(x), \phi(y), \phi(z)\}]$.

4. *Verifier:* Verifies that the answer is correct by applying the transformation, which depends on the value α.

6.3.1.3 Completeness

Certainly, the protocol is complete because the genuine prover knows both the isomorphism ϕ and the list of triples L. As it has already been pointed out, the set of triples L' to perform the series of switching transformations can be computed using the original list L and the isomorphism ϕ.

6.3.1.4 Soundness

First, it is assumed that computing L from (S_0, S_m) alone is computationally infeasible. More details on this problem are given in 5.3.

Consider an illegitimate prover who wants to cheat the verifier by convincing him of the knowledge of the secret without knowing it. On each round, he can decide one of the following actions:

1. Knowing the pair (S_0, S_m), the rogue prover is able to generate a random isomorphism ϕ' and perform it on both, S_0 and S_m. If the verifier challenges with $\alpha = 0$, then he can provide the applied isomorphism. However, if the prover is challenged with $alpha = 1$, he is required to provide the list L. Unknowing the list L, it will be impossible for the rogue verifier to provide L'.

2. A second approach is to select random triples $L' = [\{x', y', v'\}]$ and perform the series of switching transformations on an arbitrary STS T'_0. If the verifier challenges with $\alpha = 1$ then the rogue prover will be able to provide a series L' to transform T'_0 into T'_m. However, if the challenge happens to be $\alpha = 0$, he will be unable to provide a suitable isomorphism transforming $T'_0 = \phi(S_0)$ and $T'_m = \phi(S_m)$. Note that even if T'_0 is computed using a random isomorphism $\phi : S_0 \to T'_0$, it is still unfeasible to get $T_m = \phi(S_m)$.

Then, the protocol is sound because with k rounds, the probability of k correct replies is 2^{-k}, and it tends to zero as k increases, giving the rogue prover an insignificant probability of succeeding.

6.3.1.5 Zero-knowledge

To verify that the protocol is in fact zero-knowledge,

Consider a verifier that wants to obtain additional information, other than the fact that the prover knows the list L. In a round of the interaction, he can either know an isomorphism $\phi : (S_0, S_m) \to (T_0, T_m)$ or the list L'. In the first case, the only way to get the original list of triples L is by computing a series of triples $[\{x'_i, y'_i, v'_i\}]_{i=1}^m$ used to transform T_0 into T_m to obtain $[\{\phi^{-1}(x'_i), \phi^{-1}(y'_i), \phi^{-1}(v'_i)\}]_{i=1}^m$. We have argued that addressing this problem directly is unfeasible, and interacting until getting a repeated pair (T_0, T_m) requires that the same isomorphism ϕ is chosen again.

This is, consider that the verifier always challenges with $\alpha = 0$ and keeps track of the isomorphism ϕ_i for a series of rounds $i = 1, \ldots, r$. This problem can be expressed by means of the birthday paradox: the malicious verifier needs to interact i, j

6.3.2 Cryptanalysis

The proposed algorithm relies in the security of two main problems: the isomorphism of Steiner triple systems and the inversion of a composition of cycle switches. However, an attacker may consider a slight variation of the isomorphism problem:

Problem: Given four STS(n) S_0, S_1, T_0, T_1 such that a permutation $\phi : X \to X$ induces isomorphisms $\phi : S_0 \to T_0$ and $\phi : S_1 \to T_1$, find that permutation.

Though slightly simpler than the Isomorphism Problem, there is no immediate modification of the algorithm proposed in [39] to solve the isomorphism problem in $n^{\log n}$ steps. However, to ensure that the protocol remains secure, both couples of isomorphic instances must be difficult to address. Two main results discussed in Chapter 5 and Section 3.2.2 will help to provide an appropriate construction of such pairs: the analysis of difficult instances of isomorphism problems and the effect of the switching transformation on cycle graphs. For the second issue, triples such that the number of connected components in each cycle graph is still big after the switching transformation are selected. Knowing that this selection is possible is key in the protocol construction. Nonetheless, the number of suitable instances for a given n remains unknown. For instance, the number of non-affine Hall triple systems of a given order 3^k is still unknown for $k > 8$. Future research in this direction is required to provide a complete construction.

In regards to the problem of inverting switching transformations, it is conjectured that any isomorphism class of STS(n) can be reached by applying switching transformations. Since the amount of STS(n) isomorphism classes grows exponentially as n grows, it is difficult to keep track of all possible sequences of switching cycles. Additionally, the sequence of triples that can be chosen is given by

$$\left[\binom{n}{3} - \frac{n(n-1)}{6}\right]^m = \left[\frac{n(n-1)(n-3)}{6}\right]^m$$

which grows exponentially in the number transformations performed. Thus, it is also unfeasible to perform an exhaustive search of such transformations. Then the problem of finding the secret information from the public pair (S_0, S_m) is not computationally tractable.

6.4 Implementation issues

The main goal of this work is to study and define precise characteristics to propose difficult instances of the isomorphism problem in Steiner triple systems, which has

been successfully addressed in Chapter 5. Some zero-knowledge proof systems were adapted and proposed to take advantage of the results obtained.

One of the most important issues is referent to the encoding and transmission of the information. Remind that the aforementioned protocols require interaction between two entities, who have to exchange information each round. It is possible to see that the exchange of a permutation or the lists of triples is not particularly demanding, but sending the definition of a STS. A straightforward encoding lies in the definition matrix of the associated Steiner quasigroup, requiring exactly n^2 entries. However, the commutativity of such quasigroup makes that only the upper triangular matrix (without the diagonal) is required. This reduces the number of values to be stored to $\frac{n(n-1)}{2}$. Finally, the redundancy introduced by the operation defined by the set of blocks makes that only $\frac{n(n-1)}{6}$ values are indispensable. These arguments can be seen in the following Cayley tables.

$\circ$	0	1	2	3	4	5	6
0	0	3	6	1	5	4	2
1	3	1	4	0	2	6	5
2	6	4	2	5	1	3	0
3	1	0	5	3	6	2	4
4	5	2	1	6	4	0	3
5	4	6	3	2	0	5	1
6	2	5	0	4	3	1	6

$\circ$	0	1	2	3	4	5	6
0	-	3	6	1	5	4	2
1	-	-	4	0	2	6	5
2	-	-	-	5	1	3	0
3	-	-	-	-	6	2	4
4	-	-	-	-	-	0	3
5	-	-	-	-	-	-	1
6	-	-	-	-	-	-	-

$\circ$	0	1	2	3	4	5	6
0	-	3	6	-	5	-	-
1	-	-	4	-	-	6	-
2	-	-	-	5	-	-	-
3	-	-	-	-	6	-	-
4	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-

Another possible solution to this problem is to provide a minimal defining set. This could provide an optimal encoding of such a system, but, as mentioned in 3.1.1, the reconstruction of the unique STS might pose problems since embedding a partial STS into a STS of a given size can be an NP-complete problem. Additionally, finding minimal defining sets is not an easy task. An efficient generation of defining sets that lead to reconstruct the complete system easily can be helpful.

Chapter 7

Conclusions and future work

In Chapter 5, the main goal of our research was obtained. This goal consisted in describing the characteristics that a STS must have to generate difficult instances of the isomorphism problem. The process followed directly from the improvements to the Miller's technique, proposed in Algorithms 4 and 6. The identification of such instances allows to define cryptographic applications, some of them corresponding to ZKP systems, detailed in Chapter 6.

Similar to the isomorphism problem in graphs, the isomorphism problem in STSs turns out to be tricky. Difficult instances are scarce, and difficult to identify. Furthermore, even though we have found that difficult instances exist by performing an analysis on how Miller's algorithm works, it does not guarantee that specialized algorithms cannot be proposed to deal with these special cases. Additionally, we are aware that improvements on the proposed algorithms can be done by employing more efficient implementations, including the use of more suitable languages, such as C/C++, and distributed computing.

The improvements and analysis performed in Chapter 6 leads to future research lines that can be followed to provide further improvements. In fact, the *surgery* technique for joining cycle graphs, explained in Algorithm 4, can be more selective by performing a more careful analysis of how cycle graphs are affected under the cycle switch transformation. Remind that, in order to achieve a valid result, the joining of cycles by means of the cycle switch transformation must ensure that T is isomorphic to T' (obtained in steps 4 and 9 of the algorithm). Since the cycle vector is a reliable invariant, the idea is to provide a set of transformations such that the cycle vector of T and T' are identical.

Another possible source of improvements can be found in how generators are determined in Algorithm 6. Observe that the first set of generators is obtained by

using the original idea of Miller, but starting with elements that generate "big" subquasigroups. The main source of complexity is introduced in step 5, where a second set of suitable generators must be computed. The selection must be made as defined in Algorithm 5, but having that the selection of a different vertex in a cycle graph leads to a different bijection definition. Since every possible assignation should be considered, the complexity is increased. A more refined study to discard unfeasible vertex selections could be performed to reduce this computational burden.

Another subject for future research can be determined by the search of Hall triple systems. It has been mentioned that, for $k \geq 8$, the exact number of isomorphism classes of HTSs is unknown. Two main issues stand out:

- More accurate bounds on the number of existing isomorphism classes of HTSs of order 3^k could be reached from better algorithms for deciding isomorphism, however this would undermine the strength of the proposed systems.

- If HTSs offer difficulties for the isomorphism problem, also finding accurate bounds the number of isomorphism classes will be difficult.

Knowing an estimate of isomorphism classes is essential since very few of them would lead to an insecure protocol. In this same direction, the improvement of Hill-climbing algorithms, suitable for big instances is required.

More cryptographic uses can be considered by examining the problems discussed in this work, for instance, the difficulty of inverting a composition cycle switches, and the NP-complete problem of embedding a partial STS into a full system.

Finally, in the near future, we expect to publish the results obtained from this work. Also, it is desirable to adapt the parallel algorithm proposed in [8] to the techniques here developed.

Bibliography

[1] Babai, L. Graph isomorphism in quasipolynomial time. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing* (Cambridge, MA, USA, June 2016), ACM Press, New York, NY, USA, pp. 684–697.

[2] Bellare, M., Micali, S., and Ostrovsky, R. Perfect zero-knowledge in constant rounds. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing* (Baltimore, Maryland, USA, May 1990), ACM Press, New York, NY, USA, pp. 482–493.

[3] Beneteau, L. Topics about moufang loops and hall triple systems. *Simon Stevin 54*, 2 (1980), 107–124.

[4] Bours, P. A. H. On the construction of perfect deletion-correcting codes using design theory. *Des. Codes Cryptography 6*, 1 (July 1995), 5–20.

[5] Codenotti, P., Katebi, H., Sakallah, K. A., and Markov, I. L. Conflict analysis and branching heuristics in the search for graph automorphisms. In *Proceedings of the 25th International Conference on Tools with Artificial Intelligence of the IEEE* (Herndon, VA, USA, November 2013), IEEE Computer Society, pp. 907–914.

[6] Colbourn, C. Embedding partial steiner triple systems is NP-complete. *Journal of Comb. Theory A 35*, 1 (1983), 100–105.

[7] Colbourn, C., and Rosa, A. *Triple Systems*. Oxford mathematical monographs. Clarendon Press, 1999.

[8] Colbourn, C. J., and D R Stinson, L. T. A parallelization of miller's $n^{\log n}$ isomorphism technique. *Information Processing Letters 42* (1992), 223–228.

[9] Colbourn, C. J., and Dinitz, J. H. *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2006.

[10] Colbourn, C. J., Dinitz, J. H., and Stinson, D. R. *Applications of Combinatorial Designs to Communications, Cryptography, and Networking*. Lond. Math. S. Cambridge University Press, 1999, pp. 37–100.

[11] Colbourn, M. J., and Colbourn, C. J. Concerning the complexity of deciding isomorphism of block designs. *Discrete Appl. Math. 3*, 3 (1981), 155 – 162.

[12] Cox, D. A., Little, J., and O'Shea, D. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, 3 ed. Undergraduate Texts in Mathematics. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[13] Ding, J., Gower, J. E., and Schmidt, D. S. *Multivariate Public Key Cryptosystems*, 1 ed., vol. 25. Springer-Verlag, Berlin, Heidelberg, 2006.

[14] Ding, J., and Schmidt, D. Rainbow, a new multivariable polynomial signature scheme. In *Proceedings of the Third International Conference on Applied Cryptography and Network Security* (New York, NY, USA, June 2005), Springer, Berlin/Heidelberg, Germany, pp. 164–175.

[15] Doyen, J., and Wilson, R. M. Embeddings of steiner triple systems. *Discrete Math. 5*, 3 (July 1973), 229–239.

[16] Fisher, R. A. The arrangement of field experiments. *Journal of the Ministry of Agriculture of Great Britain* (1926), 503–513.

[17] Fisher, R. A., and Yates, F. *Statistical tables for biological, agricultural and medical research*, 3 ed. Oliver & Boyd, 1949.

[18] Forbes, A. D., Grannell, M. J., and Griggs, T. S. Configurations and trades in steiner triple systems. *Australas. J. Combin. 29* (2004), 75–84.

[19] Forbes, A. D., Grannell, M. J., and Griggs, T. S. On 6-sparse steiner triple systems. *J. Comb. Theory Ser. A 114*, 2 (Feb. 2007), 235–252.

[20] Gibbons, P. B., and Mathon, R. The use of hill-climbing to construct orthogonal steiner triple systems. *J. Comb. Des. 1*, 1 (1993), 27–50.

[21] Goldreich, O. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, June 2008.

[22] Goldreich, O., Micali, S., and Wigderson, A. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM 38*, 3 (July 1991), 690–728.

[23] González Fernández, E., Morales-Luna, G., and Sagols Troncoso, F. *A Zero-Knowledge Proof Based on a Multivariate Polynomial Reduction of the Graph Isomorphism Problem*. Preprints. 2018.

[24] Grannell, M., Griggs, T., and Murphy, J. Switching cycles in Steiner triple systems. *Utilitas Math. 56* (1999), 3–21.

[25] Grannell, M., Griggs, T., and Whitehead, C. The resolution of the anti-pasch conjecture. *J. Comb. Des. 8* (01 2000), 300–309.

[26] Grannell, M. J., and Griggs, T. S. Configurations in steiner triple systems. In *Ch. CRC Res. Notes*. CRC Press, 1999.

[27] Hall, J. I. On the order of Hall triple systems. *J. Combin. Th. 29* (1980), 261–262.

[28] Hilmar, J., and Smyth, C. Euclid meets bzout: Intersecting algebraic plane curves with the euclidean algorithm. *Am. Math. Mon. 117*, 3 (2010), 250–260.

[29] Junttila, T., and Kaski, P. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proceedings of the Meeting on Algorithm Engineering & Experiments* (New Orleans, Louisiana, 2007), SIAM, Philadelphia, PA, USA, pp. 135–149.

[30] Kaski, P., Mäkinen, V., and Östergård, P. R. J. The cycle switching graph of the Steiner triple systems of order 19 is connected. *Graph. Combinator. 27*, 4 (July 2011), 539–546.

[31] Kaski, P., Östergård, P. R. J., Topalova, S., and Zlatarski, R. Steiner triple systems of order 19 and 21 with subsystems of order 7. *Discrete Math. 308* (July 2008).

[32] Kirkman, T. P. Lady's and gentleman's diary, 1850.

[33] Kreher, D. L., Stinson, D. R., and Veitch, S. Block-avoiding point sequencings of directed triple systems. arXiv:1907.11186, 2019.

[34] Lin, S., and Kernighan, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res. 21*, 2 (Apr. 1973), 498–516.

[35] Lindner, C. C., and Rodger, C. A. *Design Theory*, 2 ed. Chapman & Hall/CRC, 2008.

[36] Ling, A. C. H., Colbourn, C. J., Grannell, M. J., Griggs, T. S., Ling, A. C. H., Colbourn, C. J., Usa, T. V., and Griggs, T. S. Construction techniques for anti-pasch steiner triple systems. *J. London Math. Soc* (2000), 641–657.

[37] Matsumoto, T., and Imai, H. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Advances in Cryptology — EUROCRYPT '88* (Davos, Switzerland, May 1988), Lect. Notes Comput. Sc., Springer, Berlin/Heidelberg, Germany, pp. 419–453.

[38] McKay, B. D., and Piperno, A. Practical graph isomorphism, II. *J. Symb. Comput. 60* (2014), 94 – 112.

[39] Miller, G. L. On the $n^{\log_2 n}$ Isomorphism Technique (A Preliminary Report). In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing* (San Diego, California, USA, May 1978), ACM Press, New York, NY, USA, pp. 51–58.

[40] Murphy, S., and Paterson, M. B. A geometric view of cryptographic equation solving. *J. Math. Cryptol.* (2008), 63–107.

[41] Näslund, M. On steiner triple systems and perfect codes. *Ars Comb. 53* (1999).

[42] Nemhauser, G. L., and Wolsey, L. A. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[43] Neuen, D., and Schweitzer, P. Benchmark Graphs for Practical Graph Isomorphism. In *Proceedings of the 25th Annual European Symposium on Algorithms* (Dagstuhl, Germany, 2017), vol. 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 60:1–60:14.

[44] Patarin, J. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *Proceedings of the International Conference on the Theory and Application of Cryptographic — EUROCRYPT '96* (Saragossa, Spain, May 1996), Springer, Berlin/Heidelberg, Germany, pp. 33–48.

[45] Pérez Villegas, E. Plataforma de experimentación criptográfica basada en Geometría Algebraica. Master's thesis, Computer Science Department, CINVESTAV-IPN, Mexico City, 2017.

[46] Petelczyc, K., Pramowska, M., Pramowski, K., and ynel, M. A note on characterizations of affine and hall triple systems. *Discrete Math. 312* (08 2012), 23942396.

[47] Pflugfelder, H. O. *Quasigroups and Loops: Introduction.* Sigma series in pure mathematics. Heldermann Verlag Berlin, 1990.

[48] Sagols, F., Morales-Luna, G., and González Fernández, E. Steiner triple systems and zero knowledge protocols. In *Actas de la XV Reunión Española sobre Criptología y Seguridad de la Información — RECSI2018* (Granada, Spain, 2018), pp. 18–21.

[49] Sakalauskas, E. The multivariate quadratic power problem over $\mathbb{Z}_n$ is NP-complete. *Inf. Technol. Control 4*, 1 (2012), 33–39.

[50] Seberri, J., and Street, A. P. Strongbox secured secret sharing schemes. *Util. Math. 57* (2000), 147 – 163.

[51] Shcherbacov, V. *Elements of Quasigroup Theory and Applications.* CRC Press, 2017.

[52] Stinson, D. R. A comparison of two invariants for steiner triple systems: Fragments and trains. *Ars Combinatoria* (1983).

[53] Stinson, D. R. Hill-climbing algorithms for the construction of combinatorial designs. In *Algorithms in Combinatorial Design Theory*, 1 ed., vol. 26 of *Annals of Discrete Mathematics*. 1985, pp. 321–334.

[54] Stinson, D. R., and Vanstone, S. A. A combinatorial approach to threshold schemes. *SIAM J. Discret. Math. 1*, 2 (May 1988), 230–236.

[55] Stinson, D. R., and Wei, Y. J. Some results on quadrilaterals in steiner triple systems. *Discrete Math. 105*, 1-3 (1992), 207–219.