DOCTOR OF PHILOSOPHY

Compression-based Methods for the Automatic Cryptanalysis of Classical Ciphers

Al-Kazaz, Noor

Award date:
2019

Awarding institution:
Bangor University

Link to publication

PRIFYSGOL BANGOR UNIVERSITY

School of Computer Science and Electronic Engineering
College of Environmental Sciences and Engineering

Compression-based Methods for the Automatic Cryptanalysis of Classical Ciphers

Noor R. Al-kazaz

Submitted in partial satisfaction of the requirements for the
Degree of Doctor of Philosophy
in Computer Science

Supervisor Dr. William J. Teahan

May 2019

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Dr. William Teahan for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me throughout my research and when writing this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study. Also, I would especially like to thank Dr. Sean Irvine, who is a senior research and development engineer for Real Time Genomics company in New Zealand, for collaboration on several of the joint publications.

I would like to thank the Iraqi Ministry of Higher Education and Scientific Research (MOHESR) and Baghdad University for sponsoring and supporting my Ph.D study.

My sincere thanks also goes to my family: my dear parents, my beloved husband and to my sisters for their moral support throughout my time at Bangor, for their love and for having faith in me.

I owe immense thanks to a very special person, my husband, Ali, for his encouragement and understanding during my pursuit of Ph.D degree. He was always available, even when I thought that it is impossible to continue, he helped me to keep things in perspective. I greatly value his contribution and deeply appreciate his belief in me. Many thanks go to my precious kids Danya and Mustafa for their great support and encouragement, for being there for me in both the good and hard times through this rough journey and for cheering me up and putting a smile on my face when I most needed. I really appreciate the patience they showed during all these years of study.

Last but not least, I would like to thank my fellow colleagues: firstly, Nadim Ahmed, for his unfailing assistance, his patience and support in overcoming numerous obstacles I faced throughout my research, and secondly to Maha Alamri for the support she gave me throughout my research. Special thanks go to both of them for their feedback, cooperation and of course friendship.

Abstract

The study documented in this thesis investigates the effectiveness of compression in the field of cryptanalysis, specifically for the automatic cryptanalysis of classical ciphers, initially for the English language. Several new compression-based cryptanalysis methods are developed against these ciphers.

The new methods use the well-known compression scheme—prediction by partial matching (PPM)—and have been applied to automatic cryptanalysis for three main classical ciphers: simple substitution, transposition and Playfair ciphers. The extensive set of case studies adopted in this research have validated the new methods, which have proven to be very effective in the cryptanalysis of these cases with a high success rate—for substitution ciphers, 92% of the cryptograms were correctly solved with no errors and 100% with just three errors or less; a 100% decryption success rate was achieved for transposition ciphers and 87% was achieved for Playfair ciphers. This study led to the decipherment of more challenging cases, such as very short ciphertexts with no probable words. The Gzip compression scheme has also been applied to the automatic decryption of simple substitution and transposition ciphers, but the results showed that Gzip, in comparison to PPM, was not as effective. A third compressor, Bzip2, could not be used as the nature of that scheme made its use unfeasible.

The PPM compression-based cryptanalysis methods offered significant improvements in decryption accuracy in a diverse range of experiments while being computationally more efficient compared to previously published techniques. In addition, extensive investigations were conducted to determine the most appropriate type of PPM scheme to be applied in the cryptanalysis of these ciphers. These findings have highlighted why better models are of vital importance in cryptology. In particular, the study has shown how a good model of the source (i.e. the PPM compression model)–a method that shows a high level of performance when applied to different language modelling tasks–can also be effectively used in the automatic decryption of different classical ciphers.

As spaces have been traditionally omitted from ciphertext, a full cryptanalysis mechanism which also automatically adds spaces to decrypted texts, again using a compression-based approach, has also been proposed to achieve readability.

This work has also investigated whether the newly devised cryptanalysis methods are applicable to another language (specifically Arabic as it is a language non-related to English). Arabic is a rich morphological language with its own characteristics that differentiate it from other languages. The current study has specifically adapted new compression-based methods for the automatic cryptanalysis of classical Arabic ciphers (simple substitution, transposition and Playfair ciphers). Although the experiments conducted with Arabic ciphers have generally been less effective than those with classical English ciphers, excellent results have been achieved—for Arabic substitution ciphers, 72% of the cryptograms were successfully solved without any errors and over 91% with just three errors or less; a 97% decryption success rate was achieved for Arabic transposition ciphers, with this result being 73% for Arabic Playfair ciphers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Compression can be used in several ways to enhance cryptography and cryptanalysis. It has been recognized that directly encrypting redundant texts has weaknesses and is vulnerable to statistical attacks (Wilson, 1994; Irvine, 1997). One way to protect a cipher system against this type of attack is by using compression which works by removing redundancy from a text. In this study, the converse proposition investigated is that compression models can be used to break cryptosystems. Various approaches and algorithms are used for cryptanalysis. Using compression methods as one way to tackle the plaintext recognition problem is still a relatively new approach with comparatively few publications. This observation has provided the motivation for the research described in this thesis.

The current study has investigated the effectiveness of compression for the automatic cryptanalysis of classical ciphers for English and Arabic. It develops several new compression-based cryptanalysis methods against these ciphers. These methods use compression to tackle the plaintext recognition problem for cryptanalysis. They are based on using prediction by partial matching (PPM) compression models, in addition to other standard compression methods, such as Gzip, for the automatic decryption of the three main classical ciphers: simple substitution, transposition and Playfair ciphers.

The ciphertext only cryptanalysis of simple cipher systems heavily depends on the statistical features of the source language, and it is not a trivial issue to get computers performing this analysis. Although computers have been routinely used for a variety of tasks in cryptanalysis since their invention, the automatic recognition of valid decryptions has been acknowledged as a taxing problem. Several previously published cryptanalysis methods can not run without human intervention or they assume at least known plaintext because of the difficulty of quickly recognizing a correct decryption in a ciphertext only attack (Irvine, 1997). In general, a known plaintext attack is considered to be easier to develop than a ciphertext only attack. However, for many classical ciphers, there is no effective automatic known plaintext attack, nor any published automatic ciphertext only attack (Lasry, 2018).

Modern ciphers are considered to be more complicated and secure than classical systems. However, despite this fact, classical ciphers remain a problem that has defied many different automated cryptanalysis methods (Lucks, 1990). A recently published article in the magazine *Scientific American* by Professor Bauer (2017) emphasizes why classical ciphers should matter to code breakers. Bauer stated that:

> "*Whereas encryption algorithms are continually being improved, upgrades are not always automatic. There are plenty of old systems still in use. And when it comes to ciphers used by bad guys of all sorts, old can mean centuries. Some criminals believe the National Security Agency can break all of the current systems so they fall back on older, weaker ciphers … The FBI's Cryptanalysis and Racketeering Records Unit (CRRU) regularly sees Vigenère ciphers, for example, despite the fact this method goes back to the 16th century and a method for deciphering it was published in 1863 … Criminals don't just fall back on older ciphers out of paranoia; in some cases they don't have access to the technology needed to keep their encryption up to date. A great example of this is law-breakers serving time. About 80*

percent of the CRRU's workload consists of prison ciphers. Old paper-and-pencil-type ciphers are still important ... Once again, 'historical' ciphers stayed relevant."

Professor Bauer continues:

"*In the digital world and the throwaway society we've become, where only the latest and greatest devices and technology will suffice, it can be easy to think of everything in the same way. But codes and ciphers have a long track record of being recycled, and professionals must realize that what's old is new again ... Yes, technology is always changing. But there are hundreds of examples of centuries-old ciphers that still remain unsolved to this day. They illustrate that newer is not always better as well as the importance of thinking holistically when approaching the mysteries that remain before us.*"

In addition to the motivation identified in Bauer's (2017) statements above, five other main sources of motivation lie behind investigating these ciphers. Firstly, the principles of these ciphers form the basis for many of the modern cipher systems. Most good cryptographic algorithms still combine elements of substitution and transposition (Schneier, 1996; Lasry, 2018). Hence, this study's focus is on the basic building blocks presented in many modern cryptographic systems, namely, substitution ciphers and transposition ciphers. Playfair ciphers, a more sophisticated version of substitution ciphers, are also investigated.

Secondly, these classical ciphers are regarded as a good test-bed to use for trying out new ideas for cryptanalysis. To understand and examine the basics of a new idea, it is more reasonable and useful to implement that idea on simpler cipher systems before proceeding to more difficult systems. It is hoped that the knowledge obtained from these analyses will lead to successful attacks on more difficult cryptographic methods. The new compression-based approach for the cryptanalysis of the ciphers described in this thesis is the first step towards understanding how to tackle the plaintext recognition problem that arises during cryptanalysis using compression. (The plaintext recognition problem is where the correct decrypted text is identified during cryptanalysis when compared to alternative decrypted texts).

Thirdly, despite these classical ciphers often not providing much security and they can be readily broken in many cases, there are some special circumstances where this is not the case. For example, short Playfair messages without a probable word are extremely difficult, if not impossible, to break (Cowan, 2008). In addition, some of the more challenging classical ciphers and historical messages have not yet been successfully solved. Cryptanalytic techniques exist for other cases, but only for special cases such as long ciphertexts. The modern cryptanalysis of classical ciphers can help to solve these challenging cases.

Fourthly, the study of classical ciphers in education settings is often considered a major way in which to generate student interest in studying cryptology and, in general, computer science. The best introduction to and understanding of the principles of modern ciphers are often in the context of classical cipher examples. As well as providing opportunities for increased motivation and immediate rewards, classical cryptography can provide interesting code breaking exercises and challenges. Lessons can also be drawn from past failures of classical ciphersystems, such as excessive reliance on the complexity of a cryptosystem instead of focusing on its security. As modern and historical techniques often depend on using the same statistical properties, the study of the cryptanalysis of classical cipher systems can also help in gaining an understanding of classical code-breaking methods (Lasry, 2018).

Ultimately, the use of these ciphers provided the current study with the opportunity to examine if our new compression-based cryptanalysis methods are applicable to languages other than English, especially non-related languages. As a result of the analysis, we were able to explore if another language, namely Arabic, could offer more resistant encryption than English. Arabic is a language non-related to English with its own special characteristics that differentiate it from other languages. For these reasons, simple substitution, transposition and Playfair Arabic ciphers are examined in this research. The study of classical Arabic ciphers is the first step towards investigating and solving some historical Arabic cryptograms.

1.2 Research Questions

The primary research questions for the study documented in this thesis are as follows:

1. Can compression models be used for effective cryptanalysis? Specifically, can we develop new effective methods for the automatic cryptanalysis of simple substitution, transposition and Playfair ciphers using compression?
2. Does the prediction by partial matching (PPM) compression method perform better than other common compression methods for cryptanalysis?
3. Can the newly devised methods be applicable to a language non-related to English (specifically Arabic) and how effective are these methods?

A related secondary research question is as follows:

4. Can compression models be used effectively to achieve readability of decrypted texts for cases when spaces have been omitted from ciphertexts?

1.3 Aim and Objectives

The primary aim of this study is to develop novel cryptanalysis methods using compression; specifically, to investigate the application of the PPM compression method to the automatic decryption of different classical ciphers and to determine the effectiveness of these novel approaches. Therefore, this study's objectives in investigating the research questions are as follows:

1. Produce a literature review on the area of cryptology and compression with a specific focus on the relationship between them (see Chapter 2).
2. Develop new compression-based cryptanalysis methods for the automatic decryption of simple substitution, transposition and Playfair ciphers using PPM, and compare their effectiveness with alternative compression schemes such as Gzip and Bzip2 (see Chapters 3, 4 and 5).
3. Evaluate and validate the newly devised cryptanalysis methods using an extensive set of cryptograms, especially the more challenging cases, such as short ciphertexts, and investigate the effectiveness of these methods (see Chapters 3, 4 and 5).
4. Construct a full cryptanalysis mechanism which also automatically adds spaces to decrypted texts, again using a compression-based approach, to improve readability (see Chapters 4, 5 and 6).
5. Develop and evaluate new compression-based decryption methods adapted for the automatic cryptanalysis of Arabic ciphers: simple substitution, transposition and Playfair ciphers in Arabic, and investigate whether Arabic, which is a language non-related to English, is more difficult for cryptanalysis purposes (see Chapter 6).

As stated, our study's focus is on the more challenging cases, such as short ciphertexts with no probable words. However, advantageous cases, such as long ciphers, will also be examined.

1.4 Contributions

The study reported in the thesis has made several contributions. Substantial positive evidence is presented for the proposition that compression models can be used to break cryptosystems. This evidence comprises mainly the effectiveness of a variety of new cryptanalysis methods on several different ciphers. These methods are general enough to make them applicable to other encryption systems.

Specifically, the significant contributions of this study can be listed as follows:

1. The feasibility of using compression-based approaches, specifically the PPM compression method, in the field of cryptanalysis has been further investigated.

2. An effective new method for the automatic cryptanalysis of simple substitution ciphers using PPM compression has been devised. Results on 110 cryptograms ranging from 20 to 300 characters have shown a very high success rate with about 92% of the cryptograms correctly decrypted without any errors and 100% with just three errors or less.

3. A new PPM compression-based method for the automatic cryptanalysis and plaintext recognition of transposition ciphers has been investigated with excellent results produced. The algorithm was able to achieve a 100% success rate on different amounts of ciphertext ranging from very short messages (12) to 625 characters.

4. An effective new PPM compression-based cryptanalysis method for the automatic decryption of Playfair ciphers has been proposed and investigated. The method was tried on various cryptograms of different lengths (from as short as 60 letters up to 815) and a success rate of 87% was achieved, with 100% of ciphers of lengths over 120 letters being solved. Furthermore, successful decryption of an extended Playfair cipher for a 6×6 key matrix was achieved.

5. The most efficient PPM variants for these cryptanalysis methods have been determined. The PPM methods without update exclusions have

proven to be very effective in the cryptanalysis of all three ciphers with state-of-the-art results produced.

6. The applicability of our new compression-based cryptanalysis methods to a language other than English (Arabic) has been verified. Experimental results confirmed that these methods worked well for Arabic, a language non-related to English, while previous experiments had only been conducted for English.

7. The entropy of a computer model for Arabic text has been estimated which is 1.923 bits per character.

8. Efficient PPM compression-based cryptanalysis methods adapted for the Arabic language ciphers have been developed and promising results have been obtained. For Arabic simple substitution ciphers, the results showed that 72% of the Arabic cryptograms were successfully solved without any errors and over 91% were decrypted with three errors or less. A success rate of 97% was achieved for Arabic transposition ciphers. The results for Playfair ciphers showed that 73% of the Arabic ciphertexts were effectively decrypted, and ciphers longer than 250 letters were all solved with no errors.

9. Successful word segmentation methods using the PPM compression method have been applied to achieve readability. Experimental results showed that these methods were very effective in producing a recall and precision of over 95% for the different ciphertexts that were examined.

10. It has also been discovered that the Gzip compression method performs poorly in both decryption and segmentation processing when compared to PPM.

11. A third compressor, Bzip2, could not be used because the nature of this method makes its use unfeasible for cryptanalysis.

1.5 Thesis Outline

This thesis is organized into seven chapters. Beginning with the background and motivation, **Chapter 1** then indicates the research questions and the study's aim and objectives. Research contributions and published papers are also highlighted in this chapter. To outline the overall thesis, the contents of individual chapters are briefly reviewed:

Chapter 2 reviews the general concepts of cryptology and data compression. In this chapter classical cryptology is presented primarily. Classical ciphers are investigated and different kinds of attack are highlighted. This is followed by an introduction to text compression and its main adaptive techniques. The PPM text compression method and its different variants are also introduced in detail. The relationship between these two main approaches—cryptology and compression—is illustrated. The chapter then presents the Arabic language and its specific linguistic characteristics that distinguish it from other languages. The PPM compression method, as adapted for the Arabic language, is also described.

Chapter 3 presents the first practical part of the thesis. This chapter details the new automated cryptanalysis methods that use compression against simple substitution ciphers. The following compression methods, PPM, Gzip and Bzip2, are used as a basis for this attack. Various PPM variants and PPM's different models are investigated to verify the most effective variant to be used in the automatic decryption of this cipher. The chapter reviews previous works on the automatic cryptanalysis of simple substitution ciphers. The results show how the PPM compression model achieves a very competitive performance and can be used effectively to automatically break this kind of cipher.

Chapter 4 introduces new compression-based cryptanalysis methods for transposition ciphers. Previous attacks against this cipher are also introduced. In this chapter, different PPM models and variants for cryptanalysis are examined. The Gzip compression method is also investigated. As spaces are traditionally omitted from ciphertext, two segmentation methods, also based on using compression methods, are explored. These methods, as a second step, focus on automatically inserting spaces into decrypted texts and then ranking them to find the correct solution. Excellent results are achieved using these PPM compression-based methods.

Chapter 5 proposes a novel PPM compression-based automatic attack against Playfair ciphers. A combination of two main approaches—text compression and simulated annealing—is used in the decryption of this cipher. This chapter also reports on the application of word segmentation methods. A review of the relevant work is also provided. The results show how the proposed method is efficiently able to break this cipher and can even break shorter cryptograms.

Chapter 6 reviews simple substitution, transposition and Playfair ciphers in Arabic and introduces the automatic attacks of these ciphers. This chapter proposes new adaptations of the compression-based cryptanalysis methods of these ciphers. Different Arabic corpora of different sizes are reviewed. To measure the effectiveness of the Arabic compression models, the entropy of Arabic is calculated. Related works are also presented in this chapter. The overall results show how the newly proposed methods can break these Arabic ciphers, even though this is with slightly less efficient performance compared to the English language. The results also indicate how well the new adaptive PPM methods perform the task of Arabic word segmentation.

Chapter 7 reconsiders the overall results and their significance in relation to the original aim and objectives. The conclusions derived from this study are discussed. Directions for further research are suggested.

1.6 Publications

Parts of this thesis have already been published in one journal and two refereed conference proceedings. Of the other two journal papers, one is being accepted for publication and the other is in the process of submission (see Table 1.1).

The first of these publications is a journal paper entitled "An Automatic Cryptanalysis of Simple Substitution Ciphers Using Compression". Chapter 3 of this thesis is based on this paper. The paper has been published in *Information Security Journal: A Global Perspective*, one of the Taylor & Francis journals. A link to this paper can be found with the reference for Al-Kazaz et al. (2018b).

The first conference publication, entitled "An Automatic Cryptanalysis of Transposition Ciphers Using Compression", forms the basis of Chapter 4 in this thesis. This paper was presented at the 15th International Conference on Cryptology and Network Security (CANS2016) held in Milan, Italy, and was published as a long paper by Springer in the Lecture Notes in Computer Science (LNCS) series. A link to this paper can be found in the reference for Al-Kazaz et al. (2016).

The second conference publication is another long paper entitled "An Automatic Cryptanalysis of Playfair Ciphers Using Compression". Chapter 5 of this thesis is based on this paper which was presented at the International Conference on Historical Cryptology (HistoCrypt 2018) held in Uppsala, Sweden. The paper was published as part of the Northern European Association for Language Technology (NEALT) Proceedings Series by Linköping University Electronic Press, as a freely available Gold Open Access paper. Publications in the Linköping Electronic Conference Proceedings are ranked on the Norwegian register for scientific journals, series and publishers as Level 1 publications. A link to this paper can be found in the reference for Al-Kazaz et al. (2018a).

The second journal paper "An Automatic Cryptanalysis of Arabic Transposition Ciphers Using Compression" reflects part of the work in Chapter 6. This paper has been accepted for publication in the *International Journal of Advanced Computer Science and Applications (IJACSA)*-Volume 9, No 11 November 2018.

The third journal paper "Automatic Cryptanalysis of Classical Arabic

Table 1.1: Publications

1	**Title**	An Automatic Cryptanalysis of Simple Substitution Ciphers Using Compression
	Authors	Noor R. Al-Kazaz, Sean A. Irvine, and William J. Teahan
	In	Information Security Journal: A Global Perspective
	Year	2018
	Status	Published
2	**Title**	An Automatic Cryptanalysis of Transposition Ciphers Using Compression
	Authors	Noor R. Al-Kazaz, Sean A. Irvine, and William J. Teahan
	In	Proceedings of Lecture Notes in Computer Science by Springer (LNCS)
	Year	2016
	Status	Published
3	**Title**	An Automatic Cryptanalysis of Playfair Ciphers Using Compression
	Authors	Noor R. Al-Kazaz, Sean A. Irvine, and William J. Teahan
	In	Proceedings of the Northern European Association for Language Technology (NEALT) by Linköping University Electronic Press (ECP)
	Year	2018
	Status	Published
4	Title	An Automatic Cryptanalysis of Arabic Transposition Ciphers Using Compression
	Authors	Noor R. Al-Kazaz, and William J. Teahan
	In	International Journal of Advanced Computer Science and Applications (IJACSA)
	Year	2018
	Status	Accepted
5	Title	Automatic Cryptanalysis of Classical Arabic Ciphers Using Compression
	Authors	Noor R. Al-Kazaz, and William J. Teahan
	In	Cryptologia
	Year	2018
	Status	In progress

Ciphers Using Compression" considers parts of the work in Chapter 6. This paper is currently in progress and will be submitted to the journal *Cryptologia.*

The integration of the cryptanalysis methods covered in this thesis into CrypTool 2 is being explored. CrypTool 2 (CT2) (Esslinger, 2008) is an open-source software tool developed by the research group 'Applied Information Security (AIS)' at Kassel University in Germany for illustrating cryptanalytic and cryptographic concepts, and for educating about and experimenting with cryptologic techniques. CT2 is part of the CrypTool project (Wikipedia, nd), with CrypTool a widely used e-learning tool in the fields of cryptography and cryptanalysis.

Chapter 2

Background and Related Work

2.1 Introduction

Cryptology is crucial to ensure the protection of information. The main goal of this science is to provide secure confidential systems that guarantee data integrity and privacy. This is achieved by providing well-established cryptographic algorithms that meet the objectives of securing the information system.

Text compression involves removing redundancy from a text source by reducing the space required to store the text and the time required to transfer this text without losing any information from the original source. Compression has always been related to cryptography. Many cryptosystems can be broken by exploiting statistical properties or redundancy in the ciphertext. Clearly for this reason, compression is highly recommended before the encryption process, as it removes redundancy from a text.

The general concepts of cryptology and compression are presented in this chapter. The relationship between them is also discussed. Different cipher systems are surveyed and various text compression techniques are reviewed. This fulfils objective 1 that was listed in Section 1.3. Since experimenting with a language non-related to English is one of our main objectives (objective 5), an overview of the Arabic language is also presented.

This chapter is structured as follows. Section 2.2 focuses on cryptology while Section 2.3 targets at compression. The use of compression in cryptology is addressed in Section 2.4. Section 2.5 highlights some characteristics of Arabic and the compression method used for Arabic. A brief summary is provided in the last section.

2.2 Introduction to Cryptology

The subject of *cryptology* is the study of security. This science has always been branched into two major lines of study: *cryptography* and *cryptanalysis*. *Cryptography* is the art and science of designing and implementing security algorithms that serve as primitives to provide certain security services such as integrity, confidentiality and authentication. *Cryptanalysis* is the art and science of analysing the security algorithms and defeating their security claims.

More details about cryptology and the historical development of ciphers, different kinds of attack and various cryptosystems are reviewed in the following subsections.

2.2.1 Terminology

This section presents the terminology used later to describe the work that is subsequently presented. The message (or data) which, prior to *encoding*, contains intelligible information is called *plaintext*. The output of *encoding* or *encryption* after being transformed to a "secret" unreadable message is known as *ciphertext* or a *cryptogram*. The set of functions which maps plaintext to ciphertext is called *encryption*. The key and the reverse algorithm, which generally refer to the secret information, are known as the *decryption* process. Decrypting the ciphertext restores the plaintext. An algorithm for performing *encryption* or *decryption* is known as a *cipher*. In many systems, the encryption and decryption keys are the same. Such systems are called *symmetric*; otherwise, the system is *asymmetric*. Figure 2.1 illustrates symmetric key cryptography.

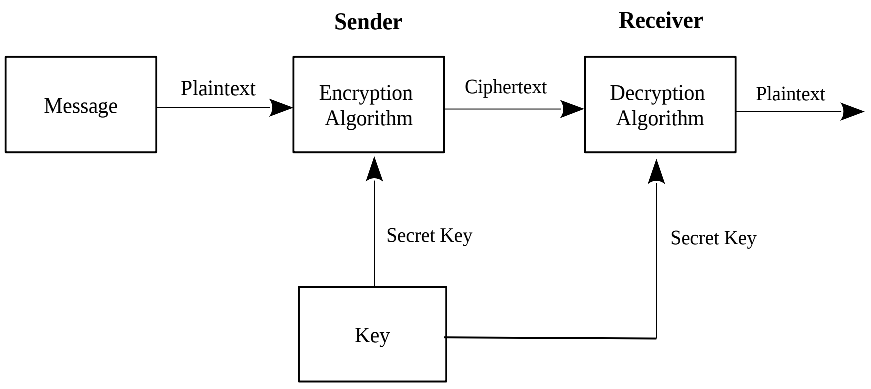


Figure 2.1: Symmetric key cryptographic system.

2.2.2 Cryptology: Where Did it Begin?

Cryptology is the science that has contributed to a variety of disciplines and cultures. The existence of this science can be traced back to the late fourth millennium BC where it was detected in one of the earliest writing systems, Sumerian scripts, in the form of logographic and syllabic units. Cryptology was also encountered around 1900 BC in Egyptian hieroglyphs. The hieroglyphic writing system combines logographic and alphabetic components that were essentially used to describe the religious literature of that era. For a long period of time, these two writing systems were considered to be an approach used for hidden communication between two different civilizations.

In the fifth century, the Egyptian Horapollo made one of the best efforts to detect these scripts. In the form of “decipherment” in the Greek text of Hieroglyphica, 200 hieroglyphic symbols were revealed (Horapollo, 1950). In addition, in the fifth, 15th and 17th centuries, collective efforts were made in the Arabic, Persian, English, Italian, Danish and German languages during various periods of time to decipher Sumerian texts.

A crucial role was played by the Greeks in constructing ancient cryptology. The use of ciphers or secret codes for the transmission of sensitive messages, whether military, diplomatic or even personal, was reported as early as the era of ancient Greece (Kahn, 1973; Bauer, 2016).

At this time, a communication channel between the sender and receiver was established as a further step to establish a level of confidentiality within cryptology. Secret messages were sent in a garbled way and remained confidential from unauthorised parties. The Caesar substitution cipher was used by the Roman general, Julius Caesar, to communicate secretly with his army during times of war. This cipher is one of the most widely known encryption techniques with each plaintext letter replaced by a letter using a certain shift from its original position in the alphabet. This cipher is referred to as a monoalphabetic substitution cipher as a fixed substitution is applied to the entire plaintext (Alkhzaimi, 2016).

Techniques for cryptanalysis developed in parallel to the evolution of cryptography. Statistical analyses were the most effective forms of attack against classical ciphers. In the 800s AD, the Arabic scientist Al-Kindi introduced the first cryptanalysis method against monoalphabetic substitution ciphers using the frequency analysis method (Broemeling, 2011). In these types of attacks, statistical analysis is performed on the number of occurrences of specific letter/word combinations. A correlation of ciphertext frequencies with plaintext frequencies and letter distributions helps the opponent to guess the original message (Alkhzaimi, 2016).

In order to increase the cryptographic security of ciphers and to overcome the limitations of single substitution, an evolution of this concept was introduced. Several substitution alphabets were used in the newly developed ciphers. Homophonic and polyalphabetic substitution ciphers are examples of these ciphers. In the homophonic substitution cipher, plaintext letters are mapped to more than one ciphertext letter. The number of potential substitutes is proportional to the frequency of the letter (the highest frequency plaintext letters are given more equivalents than lower frequency letters). The polyalphabetic substitution cipher uses multiple substitution alphabets. The best known example of this cipher is the Vigenère cipher, known as the "le chiffre indéchiffrable" and invented by Blaise de Vigenère in the 16th century. In this cipher, the letters are shifted by different amounts, usually by using a phrase or word as the encryption key. In the 19th century, the first general method of decrypting this cipher was published. Statistical analyses of digrams or trigrams in addition to the examination of repetitions were included in this attack.

More developed versions of substitution ciphers were introduced by armies during times of war and for diplomatic communications, such as the Playfair cipher. During World War I (WWI), the use of combined substitution-transposition ciphers was introduced to hide the language statistics. Cryptanalysis of the new cryptosystems has proven to be more challenging and has required more sophisticated statistical techniques. In most cases, cryptanalysis of these ciphers is extremely difficult especially for short messages with no depth.

In order to overcome the limitations of manual ciphers, several electromechanical cipher machines were introduced from the 1920s through to the early 1960s. These machines, known as rotor cipher machines, are essentially polyalphabetic substitutions which change for each letter encoded (Deavours and Kruh, 1985). The most famous rotor machine is undoubtedly the Enigma machine used extensively by Nazi Germany during WWII. In rotor machines, a number of wheels, the rotors, turn. When the first rotor completes a revolution, the second rotor is moved on by one position and so on.

The encryption machines introduced in the 1920s and 1930s also led to the invention of other cryptanalysis machines hat were used against the encryption that they produced. These machines played a major role in cryptanalysis while many cryptosystems were still decrypted by hand. The

Polish bomba and the Turing bombe are the two main examples of these machines. A team of Polish cryptographers broke a simplified version of the Enigma machine. The attack was extended to the full Enigma machine by British cryptanalysts, including Alan Turing, working at Bletchley Park, United Kingdom (UK) (Kahn, 1973).

In the 1960s and 1970s, fully electronic encryption devices and computer-based encryption were introduced. The invention of computers drove the faster cryptanalysis of the classical cryptosystems. This invention, at the same time, opened the door to the emergence of much more complex encryption systems which were impossible to implement by hand. The advent of the Data Encryption Standard (DES) is the most noteworthy event. These developments, in addition to the introduction of public key cryptography, marked the end of the era of classical cryptography (Lasry, 2018). In general, even though the principles of classical ciphers form the basis for many of the modern cryptosystems, however, modern ciphers are outside the scope of this research.

2.2.3 Attacks Against Ciphers

When decrypting a cryptosystem, the attacker can be interested in obtaining the plaintext for a given ciphertext or in detecting the key used to produce the ciphertext. As a key may have been used to encode many messages, detecting the key, in general, is the more rewarding activity.

It is usually assumed that an attacker has full knowledge of the algorithm used. This knowledge may be obtained by methods such as reverse engineering, reading the appropriate literature and so on. Identifying the details of the encryption system is quite easy if a cipher manual comes into the hands of the cryptanalyst trying to break a cipher. However, the attacker's ability may be limited in other ways which are independent of the algorithm. For example, the attacker may not be able to perform exhaustive key searches in a reasonable time owing to the limited computational ability available for their use, or they may only be able to intercept but not transmit messages.

Various kinds of attack are classified according to these restrictions.

Cryptanalytic attack can be divided into the following categories:

- **Exhaustive search**. The most obvious generic attack is the *exhaustive key search* or *brute force attack* which simply involves trying all possible keys. The difficulty of the attack corresponds to the size of the key space; thus, a very large key space can make this attack unfeasible. This attack was implemented using machines starting from the 1930s, such as in the cryptanalysis of the Enigma systems.

- **Ciphertext only attack**. The attacker only knows the ciphertext from which the plaintext or key is to be obtained. The difficulty of this attack is based on the redundancy present in the ciphertext and the available ciphertext length. This type of attack will yield no information about the plaintext (except its length), in cases where no redundancy exists in the ciphertext. Decryption methods of this type make heavy use of the source language statistics and often involve a guess of the likely parts of plaintext.

- **Known plaintext attack**. In this case, the attacker knows some plaintext with its corresponding ciphertext. This information, for many classical ciphers, allows the key to be trivially detected. This helps in reading other messages enciphered using the same or similar keys. Ranging form a piece of plaintext known to occur somewhere in the ciphertext to knowing the exact correspondence between some plaintext and ciphertext, these attacks are varied.

- **Chosen plaintext attack**. With this method, the attacker can choose his own set of plaintexts to be encoded. This enables the attacker to select a particular plaintext or plaintexts which might increase the opportunity of determining the key. In *adaptive chosen plaintext attack*, the attacker bases their next choice of plaintext on their observations from the previous encryptions.

- **Chosen ciphertext attack**. Here the attacker is able to choose ciphertexts to be decrypted in order to obtain their corresponding plaintexts. This attack is used only when the cryptanalyst wants to determine the key being used. Public key ciphers are often vulnerable to this form of attack.

- **Chosen key (related-key) attack**. In this case, the attacker knows the key in advance (Biham, 1994). During the design of a cipher, the effect of different keys on the same plaintext is investigated in this form of attack.

When evaluating the security of modern cryptosystems, it is usual to assume that a known plaintext attack and a chosen plaintext attack (in most cases) are feasible. Security always depends on several different factors while the importance of the message to be transmitted will determine the precautions to be taken. The strength of a cryptosystem is a negative quality in that security relies on the inability of attackers to find a feasible way to break it. The best way to prove the difficulty of breaking a cryptosystem is to show that its decryption operation is equivalent to solving some generally agreed computational problems that do not have a polynomial time solution (Irvine, 1997).

2.2.4 Development of Attacks Against Classical Ciphers

Cryptanalytic methods can also be divided according to their use of technology. Manual cryptanalysis is performed by hand using pen and paper only. Generally, hand approach methods are a combination of frequency analysis, pattern matching and word recognition. During WWI, these methods were the only option available at that time to use in breaking cryptosystems. The use of hand cryptanalytic methods remained extensive during WWII and continued until the invention of computers. A wealth of information about these methods has been provided by a large number of studies in the literature dating back to the early 19th and 20th centuries, as well as by National Security Agency (NSA) material that has recently been declassified. Several expositions on strategies for hand analysis have been published (Gaines, 1956; Williams, 1959; Ball, 1960; Friedman, 1976; Sacco, 1996).

In the 1930s, electro-mechanical machines and mechanized methods were developed. The Polish Bomba was developed to assist in building catalogues of the Enigma. More sophisticated machines were developed during WWII to perform processes that could not practicably be done by hand, such as the Turing bombe. Details about codebreaking machines designed before and during WWII have been published by Budiansky (2000) and Copeland (2010).

Computerized methods have been developed more recently through the adoption and use of early mainframe computers, super computers and personal computers. Starting from the 1950s, general purpose computers have been extensively used by cryptographic agencies such as the National Security Agency (NSA). Some of the material recently declassified by the NSA points to the evolution of computing simultaneously in industry and academic worlds from WWII to the 1970s, as well as the NSA's increasingly significant use of computer technology. Specifications for the newest computer systems and technology have often been driven by this cryptographic agency (Lasry, 2018). However, no information is available about the NSA's use of computers and methods to solve specific cryptosystems (Burke, 2002), and even fewer details are available about the computerized cryptanalysis of cryptograms. From the 1980s, a wide range of research about the computerized cryptanalysis of classical cryptosystems has become available to the public.

2.2.5 Classical Ciphers

Classical ciphers generally fall into two main categories: substitution ciphers and transposition ciphers. Classical ciphers are frequently used as building blocks in larger state-of-the-art cryptographic systems. Consequently, it is very important to understand the vulnerability of these simple ciphering systems, to help with building more complex ciphers (Grundlingh and Van Vuuren, 2003). Modern encryption systems have now superseded the classical systems; however, the cryptanalysis of classical ciphers remains the most popular cryptological application and implementation for metaheuristic search research. The essential concepts of substitution ciphers and transposition ciphers are still widely used today in the Advanced Encryption Standard (AES) and the International Data Encryption Algorithm (IDEA). As long as the operations and concepts of classical cipher systems are the basic building blocks of more secure modern ciphers, then classical ciphers will typically be the first ciphers considered in the case of investigating and examining new attacks (Garg and Sherry, 2005).

The rest of this section provides some examples of the classical cryptosystems used in our research. These examples illustrate different types of encoding.

2.2.5.1 Simple Substitution Ciphers

A simple substitution cipher (also called a monoalphabetic cipher) replaces each character in the plaintext with another predetermined character to form the ciphertext (Denning, 1982). Formally, let A be a plaintext alphabetic character of size n, where $A \in \{a_0, a_1, ..., a_{n-1}\}$ and C is a ciphertext alphabetic character of size n, $C \in \{f(a_0), f(a_1), ..., f(a_{n-1})\}$. Each symbol of A has a one-to-one mapping to the corresponding symbol of $C, f : A \to C$. Generally, C is a simple rearrangement of the lexical order of the symbols in A, for example:

A: `A B C D E F G H I J K L M N O P Q R S T U V W X Y Z`

C: `I R U S N V D W O X A P G T J Y B K E L M F C Z Q H`

Then the message '`CRYPTOGRAPHY DEMO`' is encoded as:

Plaintext	`CRYPTOGRAPHY DEMO`
Ciphertext	`UKQYLJDKIYWQ SNGJ`

The permutation selected represents the key.

For an alphabet with n characters, there are $n!$ possible permutations; for example, for the 26 letter English alphabet, there are 26! possible permutations or 4.03e+26. Therefore, with this large number of possibilities, finding the correct permutation through an exhaustive search is considered to be unfeasible.

This type of cryptosystem is easy to implement and to use. However, it is not difficult to crack, as it does nothing to conceal the statistical properties of the language. Hence, it does not provide much security and can be easily broken by frequency distribution analysis.

By using frequency analysis of individual letters, the cryptanalyst can readily decrypt a ciphertext manually if it uses this cryptosystem. This will still happen even if the character frequencies of the ciphertext are different from those of normal English text. With a few attempts and after trying some of the possibilities, the cryptanalyst will be able to find the correct substitution (Irvine, 1997; Eskicioglu and Litwin, 2001). Digram and trigram distributions provide more useful information that can also be accessed by the cryptanalyst. Many digrams could occur more frequently than some single letters while other digrams such as '`qj`' rarely occur in English. Typically, different languages have different letter frequencies. Thus, it is possible to determine the plaintext language before starting to decrypt the ciphertext if the ciphertext provided is of sufficient length.

Despite the fact that simple substitution ciphers are not typically used today in real-world encoding systems, many effective and secure modern ciphers use substitution ciphers in combination with other ciphers, for example, transposition ciphers, modular arithmetic, Boolean algebra and so on. This powerful combination is an important innovation as it results in a method that is stronger than its original components (Eskicioglu and Litwin, 2001).

Different cryptanalysis methods have been developed against simple substitution ciphers starting with manual methods (Ball, 1960; Friedman, 1976; Gaines, 1956) and leading to computerized methods. Various automated attacks using different searching algorithms and different scoring methods are used to break this cipher (Peleg and Rosenfeld, 1979; Lucks, 1990; Hart, 1994; Clark, 1998; Hilton, 2012; Ravi and Knight, 2009; Nuhn et al., 2013). Further description of these methods and other methods for the automatic solution of simple substitutions are discussed in detail in Section 3.2.

2.2.5.2 Transposition Ciphers

In cryptography, a transposition cipher is a method of encryption by which the content of a message is concealed by rearranging groups of letters, therefore resulting in a permutation. The concept of transposition is an essential one and has been used in the design of modern cipher systems (Stamp and Low, 2007). Originally, the message was written out into a matrix in row-order and then read out by column-order (Irvine, 1997). The technique can be expanded to d dimensions, by dividing a message into blocks or groups of fixed size d (called the period) and performing a permutation over these blocks. This permutation represents the key. The size of the key is the same as the length of the block. Generally, if $f : Z_d \to Z_d$ is a permutation over Z_d, $Z_d = \{1, ..., d\}$, then, according to f, blocks of fixed length (d characters) are encrypted by applying a permutation to the characters (Shannon, 1949; Denning, 1982). For example, if $d = 4$ and plaintext $x = 1234$, then the encrypted message (ciphertext) f might have the permutation: $f(x) : 4213$. Here, the first character in the original message is moved to the third position, the third character in the block to the fourth position, and the fourth character to the first position. Thus, the original message ‘cryptography-demo’ is encrypted as:

Position:	1234 1234 1234 1234
Plaintext:	cryp togr aphy demo
Ciphertext:	prcy rotg ypah oedm

This ciphertext is divided into blocks of four letters and in order to hide the key size (period), a stream of characters is transmitted continuously. In the case of a short block at the end, it would be encrypted by moving the letters to their relative permutation positions with dummy letters added or simply left blank.

Transposition ciphers are a class of ciphers that, in conjunction with substitution ciphers, form the basis of all modern symmetric algorithms (Giddy and Safavi-Naini, 1994). These algorithms, such as block and stream ciphers, are also used in conjunction with the above to form more complex transformations, notably for providing diffusion. Although the field of cryptology has undergone a revolution after the introduction of the asymmetric cryptographic cipher in 1976, symmetric ciphers still form the basis for secure data transmission today, owing to their superior speed and efficiency (Grundlingh and Van Vuuren, 2003).

In general, transposition ciphers are considered much harder to crack than other basic cryptosystems such as simple substitution ciphers. Many statistical tools have been developed to aid the automated cryptanalysis of simple substitution ciphers while the automatic cryptanalysis of transposition ciphers has proven more difficult. Generally, cryptanalysis of transposition ciphers is highly interventionist and demands some knowledge of the probable contents of the encrypted text to give an idea of the rearrangement order that has been used (Matthews, 1993). Related techniques for the automatic solution of this cipher are discussed in detail in Section 4.2.

2.2.5.3 Playfair Ciphers

The Playfair cipher is a symmetric encryption method which is based on bigram substitution. It was first invented by Charles Wheatstone in 1854. The cipher was named after Lord Lyon Playfair who published it and strongly promoted its use. It was considered to be a significant improvement on existing encryption methods. A key is written into a 5×5 grid and this may involve using a keyword (as in the example below). For the English language, the 25 letters are arranged into the grid with one letter omitted from the alphabet. Usually, the letter ‘I’ takes the place of letter ‘J’ in the text to be encrypted.

To generate the key that is used, spaces in the grid are filled with the letters of the keyword and then the remaining spaces are filled with the rest of the letters from the alphabet in order. The key is usually written into the top rows of the grid, from left to right, although some other patterns can be used instead. For example, if the keyword ‘CRYPTOLOGY’ is used, the key grid would be as below:

C	R	Y	P	T
O	L	G	A	B
D	E	F	H	I
K	M	N	Q	S
U	V	W	X	Z

To encrypt any plaintext message, all spaces and non-alphabetic characters must be removed from the message at the beginning, then the message is split into groups of two letters (i.e. bigrams). If any bigrams contain repeated letters, an ‘X’ letter is used to separate them. (It is inserted between the first pair of repeated letters, and then bigram splitting continues from that point). This process is repeated (as necessary) until no bigrams with repeated letters are left. If the plaintext has an odd number of letters, an ‘X’ is inserted at the end so that the last letter is in a bigram (Klima and Sigmon, 2012). For example, the message:

“*To be or not to be that is the question*”

would end up as:

“TO BE OR NO TX TO BE TH AT IS TH EQ UE ST IO NX”.

Three basic encryption rules are to be applied (Klima and Sigmon, 2012):

- If both letters of the bigram occupy the same row, replace them with letters to the immediate right, respectively, wrapping from the end of the row to the start if the plaintext letter is at the end of the row.

- If both letters occupy the same column, then replace them with the letters immediately below them. So ‘IS’ enciphers to ‘SZ’. Wrapping in this case occurs from the bottom to the top if the plaintext letter is at the bottom of the column.
- If both letters occupy different rows and columns, replace them with the letters at the free end points of the rectangle defined by both letters. Thus ‘TO’ enciphers to ‘CB’. The order is important—the letters must correspond between the encrypted and plaintext pairs (the one on the row of the first letter of the plaintext should be selected first).

Following these rules, the encrypted message would be:

“CB LI LC KG PZ CB LI PI BP SZ PI HM VD ZB DB QW”.

The Playfair cipher is one of the most well-known multiple letter enciphering systems. However, despite the high level of efficiency demonstrated by this cipher, it suffers from a number of drawbacks. The existing Playfair method is based on 25 English alphabetic letters with no support for any numeric or special characters. Several algorithms have been proposed aiming to enhance this method (Srivastava and Gupta, 2011; Murali and Senthilkumar, 2009; Hans et al., 2014). One particular extended Playfair cipher method (Ravindra et al., 2011) is based on 36 characters (26 alphabetical letters and 10 numeric characters). Here, a 6×6 key matrix was constructed with no need to replace the letter ‘J’ with ‘I’. By using the same previous keyword ‘CRYPTOLOGY’, the key matrix in this case would be:
Plaintexts containing any numerical values, such as contact number, house number or date of birth, can be easily enciphered using this extended method (Ravindra et al., 2011).

C	R	Y	P	T	O
L	G	A	B	D	E
F	H	I	J	K	M
N	Q	S	U	V	W
X	Z	0	1	2	3
4	5	6	7	8	9

Several methods for the automatic cryptanalysis of Playfair ciphers have been developed (Negara, 2012; Hammood, 2013; Stumpel, 2007; Cowan, 2008). Further description of these methods are discussed in detail in Section 5.3.

The next section provides an introduction to the area of text compression as this is an important part of our research.

2.3 Introduction to Text Compression

The primary motivation for data compression has always been making messages smaller so they can be transmitted more quickly or stored in less space. Compression is achieved by removing redundancy from the message, resulting in a more 'random' output. In practical terms, the two main classes of adaptive text compression techniques that are commonly used are: dictionary and statistical approaches (Bell et al., 1990). The dictionary approach is usually found to be faster than the statistical approach. In contrast, statistical based approaches are usually better than dictionary approaches in terms of compression rate. A third class based on block-sorting using the Burrows–Wheeler algorithm (Burrows and Wheeler, 1994) has emerged which approaches the compression rates of statistical algorithms but at much faster speeds, although not as fast as dictionary-based approaches.

2.3.1 Dictionary-Based Compression

In dictionary-based compression techniques, individual symbols (phrases) are replaced with variable length codewords (indices). By replacing long strings (phrases) with shorter codewords (indices), compression is achieved. Good compression is achieved with these techniques as many characters are represented by a single dictionary reference. There is an equivalent statistical scheme for every dictionary scheme that achieves the same compression (Bell et al., 1990). The use of these systems is still widespread as they provide a fast decompression process.

One of the most important aspects of this scheme is the construction of the dictionary. Good compression will be provided when the dictionary closely matches the text to be compressed. In this system, the length of stored phrases may be fixed or unbounded. Having a dictionary with longer phrases will offer better compression. Dictionary systems from the Ziv–Lempel family of compressors (Ziv and Lempel, 1977) are the most popular. Data are compressed in this coding scheme by providing references to the data that existed earlier (Irvine, 1997).

Gzip is one of the most important compression methods: it was written by Jean-Loup Gailly and Mark Adler, and created for the GNU project (Gzip, 2012). Gzip is a commonly used lossless compression scheme on the Internet and the Unix operating system. It uses a dictionary-based approach, which is based on the Ziv–Lempel coding scheme.

2.3.2 Block-Sorting Compression

In this scheme, which is also called the Burrows–Wheeler transform, a character string is rearranged into runs of similar characters (Burrows and Wheeler, 1994). A message is considered in blocks. For each block, the transformation which is reversible makes the transformed block easier to compress using traditional techniques such as run-length encoding.

Bzip2 is a well-known compression scheme that was written by Julian Seward (Bzip2, 2016). It is a lossless compression method which uses a block-sorting approach (the Burrows–Wheeler block-sorting compression algorithm). The compression performance of this method (Bzip2) is usually better than the Gzip; however, its speed is slower. It approaches the performance of the best compression techniques such as those produced by prediction by partial matching (PPM) (Bzip2, 2016).

2.3.3 Statistical Coding and Prediction by Partial Matching (PPM)

The best compression is achieved by another type of compressors which is based on an adaptive statistical coding approach (Bell et al., 1990). These compressors comprise two different processes: modelling and coding (Rissanen and Langdon, 1981). The model generates a probability distribution of the symbols that may occur next based on the symbols seen before in the text, while the coder is used to encode the symbol that actually occurred using this probability distribution (Teahan, 1998). Prediction by partial matching (Cleary and Witten, 1984b) and dynamic Markov compression (DMC) models (Horspool and Cormack, 1986) are two examples of this approach. However, PPM is proven to be more effective than DMC.

Prediction by partial matching (PPM) has set the performance standard in lossless compression of text throughout the past three decades (Teahan, 1998). It is an adaptive statistical coding approach which dynamically constructs and updates fixed order text compression models depending on the previous symbols being processed. The initial method was first published by Cleary and Witten (1984b). This class of text compression models performs well on English and it rivals the predictive ability of humans in comparison to other computer models (Teahan and Cleary, 1996). It has shown a high level of performance in many natural language processing tasks, such as language identification, text correction and optical character recognition (OCR) spelling correction (Teahan and Cleary, 1997)

Models that set their predictions on a few prior symbols are termed finite-context models of order k, where k denotes the number of previous symbols used. The order of the model represents the maximum context length used to predict the next symbol.

Prediction probabilities for each model are calculated from all characters or symbols that have followed every subsequence observed from 1 to length k, and from the number of times that each character has occurred. From each model, the probabilities associated with each symbol or character that has followed the preceding k characters (in the past) are estimated to predict the upcoming character. Prediction by partial matching modelling systems have been found to be very efficient at compressing English text (Teahan and Cleary, 1996).

Usually each character or symbol in the model will be encoded by using arithmetic coding depending on its associated probability (Witten et al., 1987). According to the PPM scheme, it begins from the highest context order k. Where this context predicts the upcoming symbol, then each symbol will be encoded according to its associated probability. In the case that a previously unseen character or symbol is observed in this context, the context model will not be able to encode this character and an alternative solution must be adopted. An "escape" symbol for which the probability is predicted by the PPM compression method will be transmitted to signal the encoder to switch to the next context model of order $k-1$. The operation will continue until it reaches the order in the compression model in which the upcoming character is not novel. If needed, when a completely novel symbol is encountered, the method will escape down to the $k = -1$ (order -1) default model. This is the lowest-order context in the model where all symbols will be encoded with equal probability of $\frac{1}{|A|}$, where A denotes the size of the alphabet.

By using this escape mechanism, different order models are effectively blended to 'smooth' the probability estimates. Many previous results have shown that typically no further improvement can be achieved in the compression results by increasing context lengths greater than five for English texts (Cleary and Witten, 1984b; Cleary et al., 1995b; Teahan, 1998). Further improvements can be achieved when escaping has occurred by excluding all symbols already predicted by higher-order contexts since these symbols would have already been encoded using a higher-order context if they had occurred. This mechanism is called “full exclusion”.

Moffat (1990) devised another simple mechanism that further improved results which is called “update exclusion”. This mechanism is based on how the symbol counts for each context model are updated. When encoding with update exclusions, the predicted symbol count is incremented only if it is not already predicted by any higher-order context. This means that the counts are updated only for the higher-order contexts that are actually used to predict them. Thus, the counts better reflect which symbols are likely to be excluded by the higher-order contexts. This mechanism typically improves the compression rate by up to 2% as stated by Bell et al. (1990). On the other hand, when encoding without update exclusions, all the counts for all orders of the model are updated. The counts are incremented even if they are already predicted by a higher-order context. The use of both these mechanisms is investigated in our research: PPM without update exclusions; and PPM with update exclusions (standard PPM).

2.3.3.1 Variants of PPM

Several variations of the PPM compression scheme have been invented, such as PPMA, PPMB, PPMC, PPMD, PPM* and PPMO, depending on the methods proposed for calculating symbol probabilities. Each differs by the escape method used. For example, PPMC uses escape method C, and PPMD uses escape method D. Also, the maximum order of the context models may be included when the variant is described in the literature; for example, PPMD4 refers to a fixed-order 4 PPM model using escape method D. Previous experiments showed that PPMD, in most cases, performs better than the other variants. Thus, PPMD is explored in our study in addition to PPMC which serves as a comparison. An example illustrating how these two variants work is provided below.

In the following formal description of each method, e represents the probability of the escape symbol and $p(s)$ denotes the probability for symbol s. In addition, $c(s)$ is the number of times the context was followed by the symbol s; n is the total number of times that the current context has occurred; and t denotes the total number of types.

- **PPMA**. This variant of PPM uses method A, which applies an additional count to the novel (escape) probability event. In this method, as the number of occurrences of the context increases, the escape probability decreases (Cleary and Witten, 1984a,b). For example, if a specific context has occurred seven times before, the escape probability will be equal to $\frac{1}{8}$:

$$e = \frac{1}{n+1} \qquad \text{and} \qquad p(s) = \frac{c(s)}{n+1}.$$

- **PPMB**. This variant uses method B. It classifies a symbol as being unusual or novel if it has occurred once before. This is done by subtracting one from all the counts. The idea is to filter unusual events—no prediction is made unless the symbol has occurred more than once in the current context (Cleary and Witten, 1984a,b). For example, if a context has occurred seven times with three symbols (a, b and c): a following five times, b and c once each, then the probability of a will be $\frac{4}{7}$ and there is no predictions for b and c at all:

$$e = \frac{t}{n} \qquad \text{and} \qquad p(s) = \frac{c(s)-1}{n}.$$

- **PPMC**. This variant was developed by Moffat (1990) and has become the benchmark version. The probability of this method (method C) is based on using the number of symbols that have occurred before, known as the number of types. This is similar to method B with the exception that symbols are predicted immediately:

$$e = \frac{t}{n+t} \qquad \text{and} \qquad p(s) = \frac{c(s)}{n+t}.$$

  For example, if two symbols (a and b) followed a context, twice by a and once by b, then the probability of the escape event will be $\frac{2}{5}$. In this case, 2 represents the number of types (there are two types, *a* and *b*) and 5 refers to the number of types plus the number of tokens (*a* occurs twice and *b* once, so the number of tokens is equal to 3).

- **PPMD** is another improved variant. It was first developed by Howard (1993). The PPMD variant usually shows better performance than the other PPM compression variants such as PPMA, PPMB and PPMC. This variant is similar to PPMC except that the probability of the new symbol is estimated differently. The new symbol's treatment becomes more consistent (Howard and Vitter, 1992) by adding $\frac{1}{2}$ to both the symbol and escape counts:

$$e = \frac{t}{2n} \qquad \text{and} \qquad p(s) = \frac{2c(s)-1}{2n}.$$

  For example, if a specific context has occurred three times before, with three symbols *a*, *b* and *c* following it once, then the probability for each is equal to $\frac{1}{6}$ and the escape symbol probability is $\frac{3}{6}$.

- Other variants titled **PPMP**, **PPMX**, **PPMXC** (Witten and Bell, 1991), **PPM*** (Cleary et al., 1995b), **PPMZ** (Bloom, 1998), **PPMT** (Teahan and Harper, 2001), **PPMII** (Shkarin, 2001) and **PPM-ch** (Wu and Teahan, 2005) have also been proposed.

To illustrate the process of the PPM method, Table 2.1 presents the state of the PPMC and PPMD models where $k = 2, 1, 0$ and -1 after the input string 'stressless' has been processed. For illustration purposes for this example, the highest context order is for $k = 2$. If the next symbol or character is estimated successfully by the modelling context, the probability p will be used to encode it, while c denotes the occurrence counts. Referring to the example, if the input string 'stressless' is followed by the character 'l', the probability of the prediction 'ss'→'l' in order 2 (which is $\frac{1}{2}$) would be used to encode it, requiring only one bit as a result ($-\log_2 \frac{1}{2} = 1$).

Table 2.1: PPMC and PPMD models after processing the string “stressless” with maximum order of 2.

Order k=2					Order k=1					Order k=0				Order k=-1			
Prediction			c	p	Prediction			c	p	Prediction		c	p	Prediction		c	p
PPMC																	
st	$\rightarrow$	r	1	$\frac{1}{2}$	s	$\rightarrow$	t	1	$\frac{1}{7}$	$\rightarrow$	s	5	$\frac{5}{15}$	$\rightarrow$	A	1	$\frac{1}{\lvert A\rvert}$
	$\rightarrow$	Esc	1	$\frac{1}{2}$		$\rightarrow$	s	2	$\frac{2}{7}$	$\rightarrow$	t	1	$\frac{1}{15}$				
						$\rightarrow$	l	1	$\frac{1}{7}$	$\rightarrow$	r	1	$\frac{1}{15}$				
tr	$\rightarrow$	e	1	$\frac{1}{2}$		$\rightarrow$	Esc	3	$\frac{3}{7}$	$\rightarrow$	e	2	$\frac{2}{15}$				
	$\rightarrow$	Esc	1	$\frac{1}{2}$						$\rightarrow$	l	1	$\frac{1}{15}$				
					t	$\rightarrow$	r	1	$\frac{1}{2}$	$\rightarrow$	Esc	5	$\frac{5}{15}$				
re	$\rightarrow$	s	1	$\frac{1}{2}$		$\rightarrow$	Esc	1	$\frac{1}{2}$								
	$\rightarrow$	Esc	1	$\frac{1}{2}$													
					r	$\rightarrow$	e	1	$\frac{1}{2}$								
es	$\rightarrow$	s	2	$\frac{2}{3}$		$\rightarrow$	Esc	1	$\frac{1}{2}$								
	$\rightarrow$	Esc	1	$\frac{1}{3}$													
					e	$\rightarrow$	s	2	$\frac{2}{3}$								
ss	$\rightarrow$	l	1	$\frac{1}{2}$		$\rightarrow$	Esc	1	$\frac{1}{3}$								
	$\rightarrow$	Esc	1	$\frac{1}{2}$													
					l	$\rightarrow$	e	1	$\frac{1}{2}$								
sl	$\rightarrow$	e	1	$\frac{1}{2}$		$\rightarrow$	Esc	1	$\frac{1}{2}$								
	$\rightarrow$	Esc	1	$\frac{1}{2}$													
le	$\rightarrow$	s	1	$\frac{1}{2}$													
	$\rightarrow$	Esc	1	$\frac{1}{2}$													
PPMD																	
st	$\rightarrow$	r	1	$\frac{1}{2}$	s	$\rightarrow$	t	1	$\frac{1}{8}$	$\rightarrow$	s	5	$\frac{9}{20}$	$\rightarrow$	A	1	$\frac{1}{\lvert A\rvert}$
	$\rightarrow$	Esc	1	$\frac{1}{2}$		$\rightarrow$	s	2	$\frac{3}{8}$	$\rightarrow$	t	1	$\frac{1}{20}$				
						$\rightarrow$	l	1	$\frac{1}{8}$	$\rightarrow$	r	1	$\frac{1}{20}$				
tr	$\rightarrow$	e	1	$\frac{1}{2}$		$\rightarrow$	Esc	3	$\frac{3}{8}$	$\rightarrow$	e	2	$\frac{3}{20}$				
	$\rightarrow$	Esc	1	$\frac{1}{2}$						$\rightarrow$	l	1	$\frac{1}{20}$				
					t	$\rightarrow$	r	1	$\frac{1}{2}$	$\rightarrow$	Esc	5	$\frac{5}{20}$				
re	$\rightarrow$	s	1	$\frac{1}{2}$		$\rightarrow$	Esc	1	$\frac{1}{2}$								
	$\rightarrow$	Esc	1	$\frac{1}{2}$													
					r	$\rightarrow$	e	1	$\frac{1}{2}$								
es	$\rightarrow$	s	2	$\frac{3}{4}$		$\rightarrow$	Esc	1	$\frac{1}{2}$								
	$\rightarrow$	Esc	1	$\frac{1}{4}$													
					e	$\rightarrow$	s	2	$\frac{3}{4}$								
ss	$\rightarrow$	l	1	$\frac{1}{2}$		$\rightarrow$	Esc	1	$\frac{1}{4}$								
	$\rightarrow$	Esc	1	$\frac{1}{2}$													
					l	$\rightarrow$	e	1	$\frac{1}{2}$								
sl	$\rightarrow$	e	1	$\frac{1}{2}$		$\rightarrow$	Esc	1	$\frac{1}{2}$								
	$\rightarrow$	Esc	1	$\frac{1}{2}$													
le	$\rightarrow$	s	1	$\frac{1}{2}$													
	$\rightarrow$	Esc	1	$\frac{1}{2}$													

Assume instead that the character '*t*' follows the string 'stressless'. As the order 2 model does not predict this character, the escape probability of $\frac{1}{2}$ will be encoded for this order, and the encoder will move from the order two model ($k = 2$ in the first column) down to the order one model ($k = 1$ in the second column). In this context, '*s*'→'*t*' predicts the character '*t*', with the probability $\frac{1}{7}$. Therefore, the total probability needed to encode the '*t*' character is $\frac{1}{2} \times \frac{1}{7}$, or 3.8 bits. Actually, in this context, a more accurate probability estimation is gained by noticing that the character '*t*' cannot be encoded using this context as, if it did, it would have been already encoded by the order two context. Therefore, we can exclude the already predicted symbols: this is what is termed the full exclusion mechanism, which corrects the probability for this context to $\frac{1}{6}$. Finally, the total probability will be $\frac{1}{2} \times \frac{1}{6}$ with 3.6 bits required for the compression codelength.

However, if the next character has never been seen before, such as '*m*', the escape will be repeated down through the models to the default order -1 context ($k = -1$), where all symbols or characters have equal probabilities with $\frac{1}{|A|}$ where A refers to the size of the alphabet. Supposing that the alphabet size is 256 for the English language encoded using 8-bit American Standard Code for Information Interchange (ASCII). Consequently, the total probability for encoding the '*m*' character will be $\frac{1}{2} \times \frac{3}{7} \times \frac{5}{15} \times \frac{1}{256}$, or 11.8 bits, when encoded using arithmetic coding. The full exclusion mechanism can be used to obtain a more accurate probability estimation, which will exclude characters already appearing in higher-orders. When this is applied, the new probability for the '*m*' character will be equal to $\frac{1}{2} \times \frac{3}{6} \times \frac{5}{8} \times \frac{1}{251}$, with the total codelength of 10.6 bits.

The same procedure applies to the second part of Table 2.1 when using the PPMD compression variant but using the modified symbol and escape counts as discussed above. For example, the probability of the prediction '*es*'→'*s*' in order 2 for PPMD how now changed to $\frac{3}{4}$ from $\frac{2}{3}$ and the context's escape probability has changed to $\frac{1}{4}$ from $\frac{1}{3}$ owing to the different way the probabilities are now being calculated.

The PPM method can also be applied to streams of word-based symbols as opposed to character-based streams. Several word-based systems have been proposed (Horspool and Cormack, 1992; Bentley et al., 1986; Moffat, 1989; Teahan, 1998). The word-based approach typically provides faster compression compared to the character-based models as fewer symbols are being encoded.

Similar to the previously described character-based models, word-based models use the preceding words to predict the next word using a similar PPM encoding mechanism with escapes to lower-order models. A number of methods for estimating the escape probability for the word-based models have been explained (Witten and Bell, 1991; Teahan, 1998). An escape to an order -1 word context signifies that the word needs to be encoded character by character. In this case, each symbol or character in the word (even the space character that marks the end of each word) is separately encoded. After a word has been encoded once, a word symbol associated with that word that uniquely identifies it can now be encoded instead. Essentially, the word symbols are added to an expanding alphabet of word symbols as the new words are encountered.

Previous experiments have shown that the performance of the word-based scheme degrades with higher-orders. The performance of the order 2 word bigram models is slightly worse than that of the order 1 word unigram models. Order 3 trigram word models and higher follow the same trend. Experimental results show that, for the English language, the performance of the word-based schemes is slightly better than that of the character-based ones. However, the character-based models are more economical in terms of memory space and are more easily applied to various applications in natural language processing which require the correction of character sequences such as OCR, spelling correction and cryptology (Teahan, 1998). In our work, we make use of these two models, further details of which are provided in Chapter 3.

One problem in using an adaptive compression method such as PPM is that at the beginning the models are empty with insufficient data to effectively compress the texts resulting in different permutations producing similar codelength values. To overcome this problem, a simple expedient is to prime the models using training texts that are representative of the text being compressed. In our experiments, we also use static (semi-adaptive) models that is once the models have been primed using the training texts, they are not further updated when processing the ciphertexts.

2.3.3.2 Entropy using PPM-Based Models

Both entropy and cross-entropy can be related directly to text compression. Entropy represents information content which imposes a lower bound on the number of bits per symbol that is needed to encode a long sequence of text derived from a language. Cross-entropy is an upper bound to the entropy (in this case, a particular model is used as an approximation to a language). Entropy can be used to measure how well statistical models can predict. Shannon (1951) performed experiments with humans predicting text and found that humans were able to predict English text at approximately 1 bit per character. Teahan (1998) argued that text compression provides a direct means for estimating the upper bound to the entropy. As a result, text compression can be used to directly compare performance between computer models and humans (Teahan and Cleary, 1996).

PPM codelength is the length of the compressed text, in bits, when it has been compressed using the PPM language model. It can be used to estimate the cross-entropy of the text. This is calculated according to the following formula (Chang, 2008):

$$H(S^L) = -\frac{1}{n}\sum_{i=1}^{n} \log_2 p_L(c_i|c_1,...,c_{i-1})$$

where S^L is a sequence of symbols of length n; $S^L = c_1c_2...c_n$ in language L; and p_L is a model for that language. The average number of bits required to encode the text is calculated using the model p_L, where c_i denotes the ith character of S^L.

2.4 Using Compression for Cryptology

As mentioned earlier, compression can be used in several ways to enhance cryptography. For example, many cryptosystems can be broken by exploiting statistical regularities or redundancy in the source (Lucks, 1990; Wilson, 1994). Redundancy is the bane of cryptography (Shannon, 1949). As compression removes redundancy from a source, it is immediately apparent why compression is advocated prior to encryption (Irvine, 1997). Several publications on cryptology have written about the use of this approach. Schneier (1996) devoted a page to compression, one chapter addresses compression in the book by Van Tilborg (2012), and it is described in various other publications (Gavaskar et al., 2012; Sangwan, 2012; Jasuja and Pandya, 2015; Sharma and Bollavarapu, 2015; Yilei et al., 2015; Devi and Mani, 2018). Sandoval and Feregrino-Uribe (2005) presented a hardware architecture that combines lossless compression and public key cryptography.

Considering whether data compression methods can be used to provide security has been investigated by several researchers (Witten and Cleary, 1988; Bergen and Hogan, 1993; Cleary et al., 1995a; Irvine et al., 1995; Irvine, 1997; Lim et al., 1997; Ishibashi and Tanaka, 2001; Bose and Pathak, 2006; Wang, 2006; Wen et al., 2006; Kim et al., 2007; Zhou et al., 2008; Sun et al., 2009; Wong et al., 2010; Chen et al., 2011; Duan et al., 2011; Katti et al., 2011; Kodabagi et al., 2015; Xiang et al., 2016). Most of these researchers have found that the compression schemes are insecure and some modifications have been proposed in a number of these trying to improve these methods. For example, Irvine (1997) explored the possibility of combining the two currently separate activities of compression and encryption, possibly leading to a faster and simpler communication system, through the use of compressors as cryptosystems. Different lossless compression algorithms were examined in this work. Irvine concluded that each of these algorithms has security flaws to the extent that none of these compressors should be used as security devices. However, at the same time, this work stressed that compression remains important as an adjunct to encryption, and that all critical information should be compressed before encryption to reduce the chance of ciphertext only attack.

Using compression schemes as one way to tackle the plaintext recognition problem for cryptanalysis is an approach that has resulted in relatively few publications compared to the many other methods that have been proposed for breaking ciphers. As far as we know, Irvine (1997) has been the only researcher to have previously used a text compression technique to decrypt a cryptosystem (in this case, simple substitution ciphers). Irvine used a variation of the PPM modelling system (combined with simulated annealing) for the automatic solution of simple substitution ciphers, with good results achieved compared to other methods, with 60% of ciphertexts solved without any errors, with 83% solved with less than four errors. These results were comparable to those achieved with other methods for the automatic cryptanalysis of such ciphers, and with fewer limitations. These findings have provided the motivation for the research described in this dissertation which further investigates this effective technique in order to construct new and effective cryptanalysis methods against different ciphers.

The next section presents an overview of the Arabic language. The reason is that, in a later chapter, Chapter 6, we explore the development of new automatic compression-based cryptanalysis methods specifically for the Arabic language as this is one of the research objectives (objective 5).

2.5 Introduction to the Arabic Language

The Arabic language is one of the most widely spoken languages around the world, with as many as 290 million people in Asia and North Africa and more than 400 million people around the entire world speaking Arabic. It is the fifth most spoken language (Gary and Fennig, 2018). In addition, Arabic has had a large effect on other languages like Kurdish, Persian, Pashto and Sindhi and on European languages such us Spanish, Sicilian and Portuguese (Weekley, 2012). The Arabic language is one of the Semitic languages belonging to the Afroasiatic language family. People who speak Semitic languages were the first to introduce the alphabet to the world, with the Greeks then borrowing it, after which it spread the whole world (Katzner, 2002). The Arabic language is not related to English (Comrie et al., 2009). This language is also the original language of the holy book of Muslims, "the Holy Qur'an", with a strong cultural desire seeking to preserve the vitality of the language. The importance of this language is not restricted to Muslims only, but extends to Christians as well as Jews. Some Christian and Jewish holy books and the names of some old churches were written in Arabic, the al-Muallaqah church being one example.

The Arabic language has distinct characteristics, differentiating it from other languages such as Romance languages; since for example, it is written and read from right to left. It consists of 28 consonant letters with the vowels represented by marks below and above the letters. It also has distinctive variations to represent singular, dual and plural forms and to represent male and female forms. In addition, written Arabic often exhibits triglossia with classical, modern and mixed forms frequently appearing together. These characteristics, as well as its rich morphology, present challenges for natural language processing and cryptographic systems.

In this section, a general overview of the Arabic language, with some of the structural characteristics of its printed text, and the Arab contributions to cryptology, is presented. A description of the PPM compression method for the Arabic language and calculations of the codelength metric used for plaintext recognition is also introduced.

2.5.1 Arabic Letters

Arabic letters have developed over time. the Arabic language generally consists of 28 basic letters and another eight derived letters, which have been mostly derived from the first alphabetic letter “ا”. Spaces are used to separate words. the Arabic language is a vowelised language, but it can be read and written either in a vowelised form or not, the vowelised marks being included to assist readers to read words or a whole text in the proper way. Table 2.2 presents all 36 of the Arabic alphabet letters.

Table 2.2: Arabic alphabet letters.

Basic letters	ا ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ي
Derived letters	أ إ آ ى ء ؤ ئ ة

As stated, the Arabic language is a rich morphological language (Ng et al., 2009). The morphological decomposition represented by the same Arabic word can be found in different forms in the same text. Words can take on one or more of four distinct forms. A word without any prefixes and suffixes is the first form, a word with a prefix added to it represents the second form, while the third form denotes a word with a suffix attached to it, and the fourth form is a word with both prefixes and suffixes. For example, the word “بنت”, means “girl”, while the word “البنت” with the prefix “ال” means “the girl”. The word “بنتان” with the suffix “ان” attached means “two girls” and the word “البنتان” with both the prefix and suffix “ال,ان” means “the two girls”. Furthermore, two or more prefixes and/or suffixes can be added to each word; for example, the word “والبنت” with the two prefixes “و” and “ال” means “and the girl”.

2.5.2 Arabic Encoding Methods

Each letter in the Arabic script denotes a unit of the language. Unlike the English language, there is no upper case or lower case in Arabic and each word comprising more than one letter is written joined together (cursive writing), with an exception being the letters “د ذ ر ز”, and “ا” only if they appear at the beginning of the word.

Different encoding methods are used to represent Arabic characters digitally including the ISO 8859-6 standard, Windows-1256 and UTF-8 encoding. UTF-8 encoding has been the pre-eminent character encoding scheme for the World Wide Web, with 89.8% of Web pages using it for encoding (W3Techs, 2013). UTF-8 encoding is defined by the Unicode standard and it is a compromise encoding method. It can be compact (with only one byte) like ASCII to represent an English character, but it can also represent any Unicode character with more than one byte (two to four bytes). The efficiency and compatibility shown by UTF-8 encoding for both ASCII text and Unicode scripts (that need more than one byte to represent each character, such as Chinese, Arabic and Japanese) have given it priority in many applications, websites and operating systems (Alhawiti, 2014).

2.5.3 Arabic Text Characteristics

This section provides a simplified presentation of the frequency statistics for both Arabic characters and words. The main purpose of this work is to investigate the number of times that each character has occurred in a text. Each character's percentage can be calculated using this formula:

$$character\ percentage = \frac{number\ of\ times\ each\ character\ has\ occurred}{total\ number\ of\ characters} \times 100.$$

For example, in the following sentence "تأتي على قدر الكرام المكارم", which comprises five words and 23 letters (excluding spaces), the letter "ا" occurs four times while the letter "ت" occurs twice. Therefore, the character percentage of these two letters will be (4/23)×100 =17.39% and 8.70%, respectively, according to the previous formula. The Arabic character percentage of the large Mixed Arabic corpus (further details concerning this corpus are described in Section 6.2) is explored in Table 2.3. According to this table, we can see that the characters "ا,ل ,ي,م" in addition to "spaces" (A, L, I and M in Roman alphabet characters), show the highest frequency percentage; in contrast, the letters "ظ" and "غ" show a lower frequency.

The secondary purpose here is to investigate the number of times that each n-graph occurs in an Arabic text. Table 2.4 presents the 30 most frequent n-graphs from statistics obtained from the large Mixed Arabic corpus. These statistics indicate that the occurrence of many bigraphs is much more frequent than for some single characters (unigraph), for example, “ض, غ” and “ظ”. For about 21% of the corpus, characters are represented by the top 30 bigraphs, with evidence that the Arabic words contain many repeated bigraphs. On the other hand, the top 30 trigraphs and 4-graphs represent 6.1% and 1.4% of the corpus.

A third purpose is to investigate the number of times that each word occurs in a text and to explore its frequency statistics. Arabic words are segmented using spaces with Arabic language considered to be a segmented language. As shown in Table 2.5, the 30 most frequent words represent 19% of the total number of all words in the large Mixed Arabic corpus. Previous experiments have shown that longer Arabic word lengths are used in modern texts than in classical texts (Alhawiti, 2014).

Table 2.3: Character frequency statistics for the large Mixed Arabic corpus.

Ranking	Character	Frequency	Char.%	Ranking	Character	Frequency	Char.%
1	space	50068492	18.12	20	ح	4073862	1.47
2	ا	32428109	11.73	21	ج	3124170	1.13
3	ل	26533527	9.60	22	ى	2270425	0.82
4	ي	15669503	5.67	23	إ	2103371	0.76
5	م	15002587	5.43	24	ص	2041576	0.74
6	ن	13388995	4.84	25	ش	1959720	0.71
7	و	12364000	4.47	26	ذ	1955052	0.71
8	ر	9616532	3.48	27	خ	1889753	0.68
9	ت	8944004	3.24	28	ط	1616842	0.59
10	ب	8907282	3.22	29	ث	1422756	0.51
11	ع	8012969	2.90	30	ز	1296130	0.47
12	ه	7314024	2.65	31	ض	1285156	0.47
13	د	6896282	2.50	32	غ	890614	0.32
14	ف	5976089	2.16	33	ئ	819141	0.30
15	ة	5832666	2.11	34	ء	762100	0.28
16	أ	5634256	2.04	35	ظ	569376	0.21
17	ق	5306169	1.92	36	ؤ	301583	0.11
18	س	5237184	1.89	37	آ	287076	0.10
19	ك	4573856	1.65				

Table 2.4: N-graph frequency statistics for the large Mixed Arabic corpus.

Ranking	Bigraphs	Frequency	%	Trigraphs	Frequency	%	4-graphs	Frequency	%
1	ال	13813035	4.998	الم	2202894	0.797	الله	457753	0.166
2	لم	3070976	1.111	الأ	965693	0.349	الأم	269599	0.098
3	وا	2159228	0.781	وال	950919	0.344	عليه	259414	0.094
4	ما	2136023	0.773	الت	883656	0.320	المت	241614	0.087
5	لا	2067632	0.748	على	681822	0.247	العا	208514	0.075
6	في	2042190	0.739	الع	668007	0.242	التي	203404	0.074
7	من	2034040	0.736	الل	621789	0.225	المس	202942	0.073
8	ان	1890177	0.684	قال	567484	0.205	الدو	201791	0.073
9	لي	1599212	0.579	الح	564649	0.204	لدول	179130	0.065
10	ية	1573396	0.569	الا	542462	0.196	الذي	173196	0.063
11	عل	1572128	0.569	إلى	535931	0.194	المع	172629	0.062
12	لى	1451793	0.525	الس	505206	0.183	المن	164416	0.059
13	ين	1424476	0.515	الد	489851	0.177	محمد	149102	0.054
14	ات	1364743	0.494	بال	481989	0.174	والم	138154	0.050
15	نا	1344070	0.486	لله	473191	0.171	وكان	131793	0.048
16	ول	1314876	0.476	كان	466279	0.169	النا	130967	0.047
17	ها	1292222	0.468	الن	457210	0.165	لعام	127967	0.046
18	را	1239435	0.448	الق	454139	0.164	السل	126343	0.046
19	ار	1194421	0.432	الب	453178	0.164	المو	114558	0.041
20	لت	1187878	0.430	علي	424796	0.154	الما	113851	0.041
21	بن	1175349	0.425	ليه	387279	0.140	القا	112495	0.041
22	أن	1175147	0.425	يدا	381336	0.138	لمتح	107738	0.039
23	با	1139828	0.412	الإ	380671	0.138	فقال	107233	0.039
24	ام	1133952	0.410	الج	369063	0.134	متحد	106746	0.039
25	له	1126393	0.408	ذلك	323774	0.117	الدي	106094	0.038
26	لأ	1124790	0.407	الف	316654	0.115	تحدة	105385	0.038
27	قا	1103885	0.399	الر	315786	0.114	الات	103763	0.038
28	لل	1090113	0.394	الو	306671	0.111	المر	101789	0.037
29	ري	1020244	0.369	لأم	290369	0.105	الأو	100990	0.037
30	يا	974597	0.353	هذا	289711	0.105	الإن	99889	0.036
Total		56836249	20.563		16752459	6.061		4819259	1.744

2.5.4 The Arabic Origins of Cryptography

In this section, we explore the Arabic origins of cryptography. We wish to emphasize some important early contributions to the field, and why continued research into Arabic cryptography is important in light of this historical context since we feel that research in this area is under-represented in the literature. In his famous book, *The Codebreakers*, David Kahn (1967) states that "Cryptology was born among the Arabs". In fact, Arab cryptology is older than Kahn's account indicates, and Arab contributions in this field are more broad than already detailed. Newly discovered old documents demon-

Table 2.5: Word frequency statistics for the large Mixed Arabic corpus.

Ranking	Word	Frequency	Word%	Ranking	Word	Frequency	Word%
1	في	1333465	2.663	16	و	180100	0.360
2	من	1180861	2.358	17	عبد	165011	0.330
3	بن	791876	1.582	18	كان	163659	0.327
4	على	641074	1.280	19	هذه	152062	0.304
5	إلى	526264	1.051	20	ثم	143260	0.286
6	أن	502106	1.003	21	او	137709	0.275
7	الله	410227	0.819	22	محمد	135957	0.272
8	عن	381667	0.762	23	له	129812	0.259
9	قال	302264	0.604	24	أبو	128512	0.257
10	ما	274505	0.548	25	الذي	128352	0.256
11	لا	269069	0.537	26	أبي	120425	0.241
12	هذا	233672	0.467	27	مع	120329	0.240
13	ذلك	217670	0.435	28	ابن	119567	0.239
14	التي	189731	0.379	29	لم	114368	0.228
15	عليه	187072	0.374	30	وكان	107100	0.214
				Total		9487746	18.950

strate that one of the important origins of cryptology is ascribed to the eighth century Arab researcher Al-Kindi. His investigation of cryptology is one of the oldest available books on the subject, although his antecedent Al-Khalil (718-786) is reported to have written Kitab Al-Mu'amma "The Book of Cryptographic Messages" about a century sooner. Unfortunately this book has not been found (Al-Kadit, 1992).

The word 'cipher' derives from the Arabic word sifr for the digit "zero" (0), which developed into European technical terms that mean encryption (Al-Kadit, 1992). Kahn (1967) stated that "Between AD 800 and 1200 Arab scholars enjoyed a vigorous period of intellectual achievement. At the same time, Europe was firmly stuck in the Dark Ages. While Al-Kindi was describing the invention of cryptanalysis, Europeans were still struggling with the basics of cryptography".

The 8th century heralded the golden age of Islamic civilization. Both the sciences and arts prospered in equal measure. The heritage of Arabic scientists is noticeable from the number of words that infuse the lexicon of modern science, with ‘algebra’ and ‘zenith’ being two examples. Organised and wealthy society at the time depended on an effective system of administration, and consequently the administrators depended on secure communication achieved through the use of encryption (Singh, 2000). Among the most important factors that led to progress in Arab cryptology were linguistic studies, translation, administrative sciences, public literacy and advanced mathematics.

Cryptography has been practised to disguise texts since antiquity by various civilizations, including the ancient Egyptians, Mesopotamians, Greeks, Romans, Chinese and Indians. However in none of these was there any cryptanalysis; cryptography only existed (Kahn, 1967). Cryptology, the science of making ciphers (cryptography) and decrypting them (cryptanalysis), became more prominent just after the ascent of the Arab Islamic empire. They broke the mono-alphabetic substitution cipher after many years of its successful use (Singh, 2000). Many Arab scholars wrote on, and surpassed in practising, both two branches of cryptology such as: Al-khalil, Al-Kindi, Ibn adlan, Ibn dunaintr and Ibn ad-duraihim (Al-Kadit, 1992). We will focus on the contribution of one of the earliest Arab cryptologists, Al-Kindi, in the following paragraphs.

Abu Yusuf Yaqub Al-Kindi, the author of one of the oldest known books on cryptology, was born in Al-Kufah around the year 801 AD. and grew up in Baghdad, where he received his education. He excelled in many fields such as: medicine, philosophy, mathematics, music, astronomy and linguistics. He was known as “The Philosopher of the Arabs”. His oldest available book on cryptology, *Risala fi Istikhraj Al-Kutub Al-Mu’amah* (“A Manuscript on Deciphering Cryptographic Messages”) was recently discovered in the Ottoman Archive in Istanbul. It was published by the Arab Academy of Damascus in 1987 (Mrayati et al., 1987). Al-Kindi (c.801-873 AD) presented a tree-diagram classification of the major types of cipher systems and described how to cryptanalyze them. He classified Arabic phonetics into consonants, long vowels and short vowels. Al-Kindi also introduced a comprehensive study of Arabic syntax and demonstrated possible and impossible letter combinations.

The above contributions are striking, but one of the most important of Al-Kindi's contributions includes the description and use of statistical techniques in cryptanalysis. He explained clearly how to use letter frequency statistics of the ciphertext in order to solve it. He also described how to find these letter frequencies (by utilizing an example of the same language) (Al-Kadit, 1992; Lasry, 2018). This method, which is called frequency analysis, led to the first great breakthrough in cryptanalysis (Singh, 2000).

Equally important is the condition he set on the length of the cryptogram under consideration. Al-Kindi showed that the texts should be long enough in order to allow letter frequency statistics to be meaningful. If the texts are short (less than one hundred letters), the decryption will be very difficult. This concept, presented by Al-Kindi over 1100 years ago, is very important in statistics today. Later in his book Al-Kindi set out the first count of Arabic letter frequencies. His example comprised "seven pages of Arabic" totalling 3667 letters. Al-Kindi also managed to analyse his results (Al-Kadit, 1992). These frequencies have been corrected latter by Ibn Dunaynir and Ibn Adlan, who worked on the text (Habeeb, 2016).

Al-Khalil's combinatorics and Al-Kindi's statistics are, most likely, one of the world's first writings in the field of linguistics. So, it can be argued that Al-Kindi is a forefather not only of cryptology but of statistics and linguistics as well (Al-Kadit, 1992).

It is worth mentioning that the sudden and quick descent of Arab civilization led to the absence of many books and manuscripts related to the subject, and also may have set back cryptology (Kahn, 1967). That quick descent had a significant negative effect on the progress of human knowledge and caused a serious delay in many other fields.

2.5.5 PPM Compression Method for Arabic

In this section, we describe the PPM method that we used for our cryptanalysis solution in order to encode Arabic text efficiently. In the PPM compressor, as stated above, the previously transmitted symbols are used to condition the probability of the next symbol. The predictions are based on simple frequency counts of the transmission to that point. The primary decision to be made is the context length that is modelled. The use of both PPM without update exclusions and PPM with update exclusions (standard PPM) for the Arabic language are investigated in this research.

To explain the process of the PPM method for Arabic, Table 2.6 illustrates the state of the PPMD model where $k = 2, 1, 0$ and -1 after the string "ستريسسليسس" has been processed. For illustration purposes for this example, the highest context order is for $k = 2$. If the next character is estimated successfully by the modelling context, the probability p will be used to encode it, while c denotes the occurrence counts. Concerning the example, if the input string "ستريسسليسس" is followed by the character 'ل', the probability of the prediction 'سس'→'ل' in order 2 (which is $\frac{1}{2}$) would be used to encode it, requiring only one bit as a result ($-\log_2 \frac{1}{2} = 1$).

Assume instead that 'ت' follows the string 'ستريسسليسس'. As the order 2 model does not predict this character, the escape probability of $\frac{1}{2}$ will be encoded for this order, and the encoder will move from the order 2 model down to the order 1 model. In this context, 'س'→'ت' predicts the character 'ت', with a probability of $\frac{1}{8}$. Thus, the total probability needed to encode the 'ت' character is $\frac{1}{2} \times \frac{1}{8}$, requiring four bits.

In order to deal with Arabic texts, in which each character needs two bytes to be represented, an adaptive PPM compression model for Arabic language has been presented by Alhawiti (2014). This method is called Character Substitution of Arabic for PPM ("CSA-PPM"). The use of this method has not only shown a considerable improvement in Arabic text compression but also for other texts that use Arabic script, such as Persian and

Table 2.6: PPMD model after processing the string "ستريسسليسس" with maximum order of 2.

Order k=2			Order k=1			Order k=0			Order k=-1		
Prediction	c	p	Prediction	c	p	Prediction	c	p	Prediction	c	p
ست $\rightarrow$ ر	1	$\frac{1}{2}$	س $\rightarrow$ ت	1	$\frac{1}{8}$	$\rightarrow$ س	5	$\frac{9}{20}$	$\rightarrow$ A	1	$\frac{1}{\lvert A\rvert}$
$\rightarrow$ Esc	1	$\frac{1}{2}$	$\rightarrow$ س	2	$\frac{3}{8}$	$\rightarrow$ ت	1	$\frac{1}{20}$			
			$\rightarrow$ ل	1	$\frac{1}{8}$	$\rightarrow$ ر	1	$\frac{1}{20}$			
تر $\rightarrow$ ي	1	$\frac{1}{2}$	$\rightarrow$ Esc	3	$\frac{3}{8}$	$\rightarrow$ ي	2	$\frac{3}{20}$			
$\rightarrow$ Esc	1	$\frac{1}{2}$				$\rightarrow$ ل	1	$\frac{1}{20}$			
			ت $\rightarrow$ ر	1	$\frac{1}{2}$	$\rightarrow$ Esc	5	$\frac{5}{20}$			
ري $\rightarrow$ س	1	$\frac{1}{2}$	$\rightarrow$ Esc	1	$\frac{1}{2}$						
$\rightarrow$ Esc	1	$\frac{1}{2}$									
			ر $\rightarrow$ ي	1	$\frac{1}{2}$						
يس $\rightarrow$ س	2	$\frac{3}{4}$	$\rightarrow$ Esc	1	$\frac{1}{2}$						
$\rightarrow$ Esc	1	$\frac{1}{4}$									
			ي $\rightarrow$ س	2	$\frac{3}{4}$						
سس $\rightarrow$ ل	1	$\frac{1}{2}$	$\rightarrow$ Esc	1	$\frac{1}{4}$						
$\rightarrow$ Esc	1	$\frac{1}{2}$									
			ل $\rightarrow$ ي	1	$\frac{1}{2}$						
سل $\rightarrow$ ي	1	$\frac{1}{2}$	$\rightarrow$ Esc	1	$\frac{1}{2}$						
$\rightarrow$ Esc	1	$\frac{1}{2}$									
لي $\rightarrow$ س	1	$\frac{1}{2}$									
$\rightarrow$ Esc	1	$\frac{1}{2}$									

Kurdish. There are two important operations in this method, which are the pre-processing and post-processing, used in conjunction with the PPM method. Each two-byte Arabic character is substituted with an equivalent number of the UTF-8 encoding scheme in the first reversible operation and, as a result, one output file is generated. In contrast with the post-processing operation, a reverse operation is performed by replacing the numbers with the original equivalent characters.

2.6 Summary

In this chapter, an overview of the historical development of cipher systems is presented. Various kinds of attacks and different cryptosystems, some modern but most classical are introduced. Modern text compression techniques are surveyed and described. This is followed by a discussion of the PPM compression method and how the codelength metric is calculated.

The use of compression for cryptology is also illustrated. Different fundamental characteristics of the Arabic language and its encoding methods are reviewed. Arabic contributions to cryptology and the PPM compression method for the Arabic language are also described in this chapter.

Chapter 3

Designing and Evaluating a New Automatic Cryptanalysis of Simple Substitution Ciphers Using Compression

3.1 Introduction

In this chapter, we propose a new compression-based approach for the automatic cryptanalysis of simple substitution ciphers with no need for any human intervention. This helps to address research questions 1 and 2 in Section 1.2, and fulfils objectives 2 and 3 that were listed in Section 1.3. This chapter considers the application of compression to tackle the plaintext recognition problem for cryptanalysis of simple substitution ciphers.

Our automatic cryptanalysis method uses a new variation of the prediction by partial matching ('PPM') text compression scheme. This work also investigates different variants of PPM to ascertain the most efficient type when applied to the problem of decrypting simple substitution ciphers automatically using compression. The use of other well known compression schemes, Gzip and Bzip2, are also explored in this chapter.

The work in this chapter has been published in 'Information Security Journal: A Global Perspective', one of the Taylor and Francis journals (Al-Kazaz et al., 2018b).

This chapter is organised as follows. Section 3.2 provides a summary of the previous research used in the solution of simple substitution ciphers. Section 3.3 motivates the use of compression as an automatic cryptanalysis method and reviews the codelength metric calculations used in our approach which is based on the PPM, Gzip and Bzip2 compression methods. The pseudo-code and the full description of our method are presented in section 3.4. Experimental results are discussed in section 3.5. The final section provides the conclusion.

3.2 Related Work

Several cryptanalysis techniques have been devised for the solution of simple substitution ciphers, starting with a number of strategies for hand analysis, leading to automated cryptanalysis methods. In this section, we will concentrate on previous approaches for automated cryptanalysis.

Typically, human experts who have experience in cryptanalysis can solve a sentence-long ciphertext in a few minutes. Many hand analysis strategies have been described (Ball, 1960; Friedman, 1976). These strategies are generally a combination of three main classes: zero order frequency analysis, a pattern matching approach and word recognition. However, none of these strategies are explicit enough to be called an algorithm. Other different solutions have been devised over the last few decades with varying degrees of success (Schatz, 1977; Anderson, 1989).

The following summarises some of the more interesting or important automatic cryptanalysis methods and their drawbacks. The purpose is to highlight the breadth of research that has previously been applied to the problem.

In particular, most of the previous attempts for automated cryptanalysis are based on two main approaches: a probabilistic approach (Peleg and Rosenfeld, 1979; King and Bahler, 1992) and a pattern matching approach (Lucks, 1990; Hart, 1994). Automatic solutions for decrypting substitution ciphers using iterative methods were presented by Peleg and Rosenfeld (1979) and King and Bahler (1992). In these methods, breaking the cipher is considered as a probabilistic problem. Joint letter probabilities are used to update symbol probabilities, and after a number of iterations, hopefully there is an improvement in the estimations that lastly lead to solve the ciphertext. In the paper by Peleg and Rosenfeld (1979), the joint letter probabilities were based on trigram frequencies. Two examples were examined and decrypted in this paper: a 996 character long ciphertext using a paragraph from a technical report and a 1149 character long ciphertext using Lincoln's Gettysburg Address. With a 400 character long ciphertext, the method (King and Bahler, 1992) was able to correctly recover 93% of the ciphertext symbols within an average execution time of 13 minutes.

On the other hand, pattern matching algorithms (Lucks, 1990; Hart, 1994) work better on short ciphertexts, but can not solve ciphertexts for which there are no words in the dictionary being used by the algorithm. They are not able to handle trivial variations, like examples with spaces removed. According to the pattern matching approach, each word in the ciphertext is structurally compared with words in a dictionary. The accuracy of this approach and the time required to break the ciphertext depends on the size of the dictionary. A dictionary size of over 19,000 entries was used, and a hundred different examples chosen at random from magazines and newspapers were examined (Lucks, 1990). A success rate of 60% was achieved; however, about 30% of the trials required further human intervention. The second algorithm (Hart, 1994) had been tried on over a hundred short ciphertexts and generally provided readable solutions in a matter of seconds. Both methods (Lucks, 1990; Hart, 1994) did not produce complete or unique solutions. That is because either there were some words that did not appear in the dictionary or multiple possibilities were deciphered.

All these previous approaches (Peleg and Rosenfeld, 1979; Hart, 1994; Lucks, 1990) deal with just twenty six alphabet English letters and consider that the spaces between words are already identified or not ciphered. In contrast, our method described in the next sections deals with twenty seven English characters (twenty six alphabetic letters and space).

Lee et al. (2006) introduced an enhanced English frequency analysis technique which uses a combination of unigram frequencies and dictionary checking. Two ciphertexts were examined, one with 9006 letters and the other with 2802 letters, and the method was able to achieve good decryption results. Another dictionary-based attack was demonstrated by Olson (2007). Twenty one cryptograms were examined and all of them were successfully solved. However, the algorithm also struggled on short cryptograms.

A genetic algorithm for the cryptanalysis of simple substitution ciphers was published by Spillman et al. (1993). A genetic algorithm is a metaheuristic that is commonly used to generate high-quality solutions to optimization and search problems. Genetic algorithms repeatedly modify a population of individual solutions. Modifications in the population basically are achieved using two main operators: mutation and crossover. At each step, the genetic algorithm chooses individuals from the current population to be parents based on their fitness as measured by a fitness function and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution.

To evaluate the quality of a key using the genetic-based cryptanalysis method (Spillman et al., 1993), a fitness function was used based on character unigram and digram English frequencies. There was no specific description about the test set characteristics that they used in this paper, and the exact key was not always found.

A simulated annealing approach was used with the evaluation based on using bigram statistics (Forsyth and Safavi-Naini, 1993). With a very long ciphertext (5000 characters), the algorithm performed quite well, but it was less efficient with shorter ciphers. Simulated annealing (inspired by a process similar to metal annealing) is a probabilistic method for approximating the global optimisation of a given function in a large search space. It is a descendant of the hill-climbing technique. This latter technique is based on starting with a random key, followed by a random change over this key such as swapping two letters, to generate a new key. If the change produces a better solution, the new key replaces the current one. This process is repeated until there are no further improvements. Simulated annealing is similar to hill-climbing with a small modification that often leads to an improvement in performance. In addition to accepting better solutions, simulated annealing also accepts worse solutions in order to avoid the local optima. This approach permits it to jump from local optima to different locations in order to find new optima.

The use of a genetic algorithm, simulated annealing and tabu search for the cryptanalysis was also presented (Clark, 1998; Garg and Sherry, 2005). Tabu search is a metaheuristic search method that takes a potential solution to a problem and check its immediate neighbors in the hope of finding an improved solution. For this purpose, the tabu search has a short-term memory system. The short-term memory system (tabu list) stores previously visited solutions and has a set of rules to prevent the reversal of recent moves and sometimes allows non-improving solutions in order to escape the local optimum.

Character unigram, digram and trigram frequencies were adapted as the basis for the fitness function by Clark (1998) and Garg and Sherry (2005). The results obtained were very similar for each of the three algorithms. For a cryptogram of 800 characters, 25 out of 27 key elements were recovered, and for a cryptogram of length 500, it was able to recover 23 keys (Clark, 1998). For a ciphertext of 200 characters, the amount of recovered keys was about 12 whereas with a 1000 character long ciphertext, about 24 keys were successfully recovered out of 27 (Garg and Sherry, 2005).

Other genetic algorithm based solutions (Grundlingh and Van Vuuren, 2003; Mudgal et al., 2017) were successfully implemented. Just one long ciphertext of 2519 characters was examined, and the fitness function was based on character unigram and bigram analysis (Grundlingh and Van Vuuren, 2003). Many previous works in this area were summarized, and the use of genetic algorithms was specifically explored by Delman (2004). Attempts to extend these works were unsuccessful. This resulted in the conclusion that the genetic algorithms approach did not merit further effort, since although the traditional cryptanalysis methods require more execution time, they were easier to implement and much more successful.

A fast automated attack using hill climbing was presented (Jakobsen, 1995). Digram frequencies were used as the basis for calculating the scoring function. With a ciphertext of 100 characters, the algorithm achieved a success rate of 50%, and with a ciphertext of 400 characters in length, a success rate of 98% was reached. The time needed to cryptanalyze a cipher ranged from half a second to two seconds.

Other attacks using Hidden Markov Models (HMM) and hill climbing were presented (Lee, 2002; Vobbilisetty et al., 2017). Lee (2002) showed that the proposed method systematically outperformed the iterative methods using character bigram models. It achieved a 95% decoding rate for cryptograms of 500 characters, whereas just 80% was achieved by the relaxation iterative methods. A 70% accuracy rate was achieved (Vobbilisetty et al., 2017) as a result of solving a ciphertext of 200 letters in length. With ciphertexts of 300 and 400 letters, a 95% accuracy rate was achieved. Ciphertexts of 1000 and 2000 characters were used in the paper by Chen and Rosenthal (2012). An accuracy rate of 93% against a 2000 character long ciphertext using bigram was achieved.

Attacks based on different local search metaheuristics were published by several researchers (Uddin and Youssef, 2006; Hilton, 2012; Corlett and Penn, 2010; Luthra and Pal, 2011; Jain et al., 2018). Character unigram and bigram statistics were both used for the evaluation function (Uddin and Youssef, 2006). Using bigram statistics resulted in a 45% success rate for a ciphertext of 100 characters, and 95% for a cipher of 400 characters. The use of character unigram, bigram and trigram statistics were investigated (Hilton, 2012). About six correct keys were recovered out of 26 for a 100 character long ciphertext, and 20 correct keys for a cryptogram with a length of 500. According to Corlett and Penn (2010), a character-level trigram model was used to rank solutions. Texts with different sizes (1000, 3500, ... 13500) were tested. With a 3500 character long ciphertext, the accuracy was 96% with an execution time of 38 minutes. For ciphertexts of 500 characters (Luthra and Pal, 2011), the researchers were able to recover 21 correct keys out of 27, and 22 correct keys from ciphertexts of 1000 characters in length. Bigrams based cost function was used in the paper published by Jain et al. (2018). 200 different ciphertexts of size 100, 200, 300,..., 800 characters were examined. For a cryptogram of 100 characters, 9 out of 26 key elements were recovered, and for a cryptogram of length 800, it was able to recover 24 keys.

Cryptanalysis methods based on using different order n-gram models and different search algorithms were proposed (Knight et al., 2006; Ravi and Knight, 2008; Nuhn et al., 2013; Kambhatla et al., 2018). For example, a cryptanalysis method using low order n-gram models (1-gram, 2-gram and 3-gram models for English) was presented (Ravi and Knight, 2008). Fifty ciphers of different lengths were examined, on a 52-letter cryptogram, with the solution from this method resulting in 21% error. With a ciphertext of 64 letters, this method gave 10% error with an average execution time of approximately 76 minutes, and 5% error for a ciphertext of 128 letters using 3-gram models.

Another method based on using high order n-gram models (3-gram, 4-gram, 5-gram and 6-gram) and a beam search to the problem of solving substitution ciphers was introduced (Nuhn et al., 2013). Beam search is a heuristic search algorithm that explores a graph by expanding the most promising node in a limited set. Beam search uses a technique like breadth-first search to build its search tree. At each level of the tree, it generates all successors of the states at the current level, sorting them in increasing order of heuristic cost. However, it only stores a predetermined number of best states at each level (called the beam size). Only those states are expanded next. The greater the beam size, the fewer states are pruned. With an infinite beam size, no states are pruned and beam search is identical to breadth-first search.

Short cryptograms (up to 256 letters) were tested by Nuhn et al. (2013). A cryptogram of length 64 was decrypted with less than 5% error with a reported decryption time of two and a half minutes using 6-gram models. On a 128-letter cryptogram, the solution from this method resulted in 0.05% error. Decipherment of simple substitution ciphers with neural language models and a beam search was published by Kambhatla et al. (2018). The method gave 0.07% error for a 64-letter cipher and 0.02% for a 128-letter cryptogram. The beam size used in these methods was very large–100,000. Other smaller sizes were also explored, however, with less efficient performance demonstrated. For example, with a 1000 beam size, the solution from the previous method resulted in about 5% error on a 64-letter cipher and about 10% for a 128-letter ciphertext.

In comparison to the all previously mentioned studies, our evaluation analyses the decryption of different ciphertext lengths, including very short cryptograms with just 20 characters, whereas the cryptograms used in most of the previous research were not less than 100 characters long. Shorter ciphers (less than 100) usually required a long time to execute with a higher error rate compared to our method. Many previous approaches (Ravi and Knight, 2008; Nuhn et al., 2013; Kambhatla et al., 2018) assumed that the space character has already been known. Nuhn et al. (2013) stated that this assumption makes the problem much easier, and previous methods showed much higher computational demands for lengths beyond 256 letters when space is not assumed to be known. In addition, the beam size used in these researches (Nuhn et al., 2013; Kambhatla et al., 2018) was very large (100,000); using smaller sizes resulted in a higher error rate. In contrast, in our proposed method, we did not assume that the space symbol was already identified. Our compression-based method deals with twenty seven English characters (twenty six alphabetic letters and space), and many different short ciphers of 20 characters or so have effectively been decrypted. Furthermore, the beam size used in our method is very small being just 500, which is much more smaller than the previously used size, and all the cryptograms of length longer than 44 have successfully been solved without any errors with state of the art results produced.

Irvine (1997) has been the only researcher to have previously used a text compression method to decrypt a cipher system (simple substitution ciphers). Irvine used a variation of the PPM modelling system combined with simulated annealing for the automatic cryptanalysis of simple substitution ciphers. Over a hundred ciphertexts were examined, and good results were achieved compared to other methods with 60% of ciphertexts solved without any errors, and 83% with less than four errors. In this chapter, we investigate more deeply the use of PPM compression method by proposing a new variation for tackling the problem of the automatic decryption of simple substitution ciphers. However, our approach uses a different search algorithm (beam-style search) and a new modified version of PPM, which achieves a high success rate close to 100%. The use of other compression schemes (Gzip and Bzip2) are also examined. In this chapter, we present different PPM compression variants and investigate which variant is the most effective when applied to the problem of automatically decrypting simple substitution ciphertexts.

3.3 Automated Cryptanalysis Using Compression

The ciphertext only cryptanalysis of simple cryptosystems relies heavily on the statistical properties of the source language. Getting computers to perform this analysis is not a trivial matter. Although computers have been routinely used for a variety of tasks in cryptanalysis since their invention, the automatic recognition of valid decryptions has remained a taxing problem (Irvine, 1997). Several previously published cryptanalysis methods can not run without human intervention or they assume at least known plaintext because of the difficulty of quickly recognizing a correct decryption in a ciphertext only attack (Schneier, 1996; Wiener, 1993). In general, a known plaintext attack is considered to be easier to develop than a ciphertext only attack. However, for many classical ciphers, there is no effective automatic known plaintext attack, nor any published automatic ciphertext only attack (Lasry, 2018).

In the cryptanalysis of classical ciphers, there are several scoring methods that are commonly used such as those based on n-gram statistics, however, the design of a new scoring function or even the selection of an appropriate one is not an easy issue. The most critical element for successful searching algorithms is the scoring function. This function evaluates the quality of each permutation (or a candidate key) and allows the search algorithm to determine whether one permutation is better than others (Lasry, 2018).

Having a computer model that is able to predict and model natural language as well as a human is critical for cryptology (Teahan, 1998). Teahan and Cleary (1996) demonstrated that the PPM modeling system has the ability to predict text in a way that is close to achieving human performance level. The essential idea of our technique is to use PPM for calculating the compression 'codelength' value of each possible permutation (which is used to measure the amount of information in each (Irvine, 1997)). Permutations with smaller codelength values help to determine better decryptions. We present how to use this to automatically and easily recognise the valid solution in a ciphertext only cryptanalysis against simple substitution systems. Also, further investigations on different variants of PPM compression method are performed in this chapter. Other compression methods (Gzip and Bzip2) as a basis method for calculating the codelength metric are also tried. This is to ascertain the most effective compression method to use to automatically break simple substitution cryptosystems.

3.3.1 PPM Compression Codelength Metric

As stated above, the fundamental concept of our cryptanalysis method is based on using a PPM compression model to calculate codelength values of each possible permutation. The 'codelength' of a permutation for a cryptogram is the length of the compressed cryptogram, in bits, when it has been compressed using the PPM language model. The hope is that the smaller the codelength value, the more closely the ciphertext resembles the text used to train the language model.

In our work, we make use of two PPM based models, one character-based and the other word-based, which provide effective results in terms of compression rate and lead to significant improvements both in terms of compression rate as published in previous experiments (Teahan et al., 2014) and in terms of the reduction in the number of decryption errors as per the experimental results discussed below. The first model, which is labeled $C|C^5$ in Table 3.1, represents an order 5 PPM character model (order 5 and order 4 models are used in our experiments) where the predictions are based on the stream of character symbols. So, the probability of S (where S is a sequence of length m characters c_i) is given by:

$$p(S) = \prod_{i=1}^{m} p'(c_i|c_{i-5}c_{i-4}c_{i-3}c_{i-2}c_{i-1})$$

where p' is the probabilities estimated by the order 5 PPM model. A maximum order of 5 is usually used in most of the experiments (Teahan, 1998; Liu et al., 2014; Almahdawi and Teahan, 2017) and order 5 has also been found effective for both English and Arabic text (Alkahtani et al., 2015). (Note: the symbol $\hookrightarrow$ in the table represents an escape).

The second model, which is labeled $W|W$, represents an order 1 PPM

Table 3.1: Models for predicting character and word streams (Teahan, 1998).

$\mathbf{C|C^5}$ **Model**	$\mathbf{W|W}$ **Model**
$p(c_i\|c_{i-5}c_{i-4}c_{i-3}c_{i-2}c_{i-1})$	$p(w_i\|w_{i-1})$
$\hookrightarrow p(c_i\|c_{i-4}c_{i-3}c_{i-2}c_{i-1})$	$\hookrightarrow p(w_i\|)$
$\hookrightarrow p(c_i\|c_{i-3}c_{i-2}c_{i-1})$	$\hookrightarrow$ Character model
$\hookrightarrow p(c_i\|c_{i-2}c_{i-1})$	
$\hookrightarrow p(c_i\|c_{i-1})$	
$\hookrightarrow p(c_i\|)$	
$\hookrightarrow p_{eq}(c_i\|)$	

word bigram model. The predictions of this model are based on the stream of word symbols as shown in the next formula:

$$p(S) = \prod_{i=1}^{m} p'(w_i|w_{i-1})$$

where p is the probability of S, S is a sequence of m words w_i and p' is the probabilities estimated by the word model (Teahan, 1998).

The compression codelength can be used to estimate the cross-entropy of the text and can be calculated according to the following formula:

$$h(S) = -\log_2 p(S) \quad = -\log_2 \prod_{i=1}^{m} p'(c_i|c_{i-5}c_{i-4}c_{i-3}c_{i-2}c_{i-1})$$

where $h(S)$ is number of bits required to encode the text.

3.3.2 Calculating Compression Codelengths Using Gzip and Bzip2

We have also investigated alternative compression methods for performing the codelength calculations—Gzip and Bzip2. The essential reason for experimenting with other compression schemes in our work is to find out which is the most effective scheme that can be used in the automatic solution of simple substitution cryptosystems using a compression based technique.

In this chapter, the calculation of the codelength metric for these two compression methods (Gzip and Bzip2) is based on using a relative entropy calculation which allows us to use ‘off-the-shelf’ software without the need to re-implement the methods themselves. The codelength metric can be calculated using the relative entropy technique by the following formula (first described in (Al-Kazaz et al., 2016)):

$$h_t = h_{T+t} - h_T$$

where T denotes the training text (which will usually be large in size), t denotes the testing text, and h_T refers to the size of the compressed file T. Essentially, the codelength for a particular compression scheme is calculated as being the difference between the compressed size of some large training text with testing text concatenated after it compared to the compressed size of just the training text by itself.

3.4 The New Method

A full description of our new method for the automated solution of simple substitution ciphers is presented in this section. The main idea of our approach is based on trying to break a cryptogram by essentially substituting one letter at a time throughout the text, starting with the most frequent. Then one of the compression methods is used to compute the codelength value used for automatically scoring the possibilities. PPMD, PPMC, Gzip and Bzip2 are the compression schemes used in our experiments.

The pseudo-code for our approach is presented in Algorithm 1. At the start (see line 1 in the pseudo code), we remove all non-alphabetic characters from the ciphertext and keep only letters and spaces (i.e.our approach processes only 27 characters). However, the methods presented here can be adapted to arbitrary alphabets (whether or not spaces are included). After that, all the remaining characters in the ciphertext are examined in order to determine frequencies, and arranged from the most frequent character to the least frequent (see line 2). The search is initialised by setting each character in the ciphertext to a special symbol (a full-stop) that is not an English alphabetic character or space (see line 3). Then by replacing one ciphertext character *cc* at a time (see lines 6 to 22) for each permutation of the ciphertext in the list 'Q1' (lines 8 to 20), it simply tries each unused character in turn as a candidate for *cc* (see lines 9 to 11). The compression codelength is calculated for each possibility using PPM, Gzip or Bzip2 (see line 12). The ciphertexts are then ranked using a sorted list according to the codelength values (see lines 13 to 17). As we find permutations with smaller compression codelength values, we are closer to finding the valid decrypt. We keep at most only the 500 best results at each stage in the sorted list. The maximum size of the sorted list provides a means for trading off between greater speed (when the size of the list is reduced) and less decryption errors (when a greater size is used). Experimental results show (see below) that a size of 500 provides a good compromise.

Our method builds up the solution incrementally, replacing one crypto character, *cc*, at a time, dealing with the most frequent *cc* first. So starting with a new cryptogram, it picks the most frequent symbol (say x) in the cryptogram (most likely this corresponds to space or perhaps the letter 'e'). It tries substituting x with one of the English alphabetic characters 'a' to 'z' or space. (Note: At this stage, all of these will be tried since the size of the alphabet, 27, is less than the maximum size of the sorted list i.e. 500). Then it picks the next most common crypto symbol to substitute, say y, for each of the previous 27 possibilities, substituting y with one of the English alphabetic characters or space (but excluding any already substituted characters). This gives 702 ($= 27 \times 26$) possibilities, so at this stage solutions start getting discarded if only a maximum of 500 possibilities are kept in the sorted list. This is repeated for each remaining character from the most frequent characters down to the least frequent character.

In order to get further improvements in our results, a word-based PPM compression system is applied to the output produced from Algorithm 1.

Algorithm 1: Pseudo code of the automatic cryptanalysis of simple substitution ciphers.

```
   Input  : ciphertext
   Output: decrypted text(s)
1  remove all non-alphabet characters–except space–from the ciphertext;
2  examine the ciphertext to create a sorted list of the zeroth order frequencies for
   the alphabet;
3  replace the characters in the ciphertext with the special symbol ‘.’;
4  initialise Q1 (list) with a modified ciphertext;
5  maximum size of Q2 (sorted list) $\leftarrow$ 500;
6  foreach crypto character ‘cc’ in the zeroth order frequent characters (starting
   from the most to the least frequent characters) do
7      Q2 $\leftarrow$ empty;
8      foreach ciphertext in Q1 do
9          foreach alphabetic and space character ‘ac’ do
10             if ‘ac’ is not used before as a replacement of the previous crypto characters
               then
11                 replace each crypto character ‘cc’ in the ciphertext with the unused
                   character ‘ac’ as a candidate for ‘cc’;
12                 calculate codelength value of the ciphertext using the PPM, Gzip or
                   Bzip2 compression model;
13                 if the size of the sorted list Q2 $<$ 500 then
14                     add the ciphertext to Q2;
15                 else if the codelength value of the last element in Q2 $>$ codelength value
                   of the current ciphertext then
16                     remove the last element in Q2;
17                     add the ciphertext to Q2;
18             end
19         end
20     end
21     replace Q1 with Q2;
22 end
23 return Q1 containing the best solutions (the ‘decrypted text(s)’);
```

The pseudo-code for this is provided in Algorithm 2. For each text in ‘Q1’, the codelength value is re-calculated using the word-based model. Then these are stored in a new list as they provide a potentially more accurate estimate of their quality (lines 2 to 5).

Algorithm 2: Pseudo code of word-based ranking algorithm.

```
Input  : the list Q1 (output from Algorithm 1)
Output: decrypted text(s)
1 maximum size of Q3 (sorted list) ← 500;
2 foreach text in Q1 do
3     calculate compression codelength value of the text using the PPM word-based
      compression method;
4     store the text in the sorted list Q3;
5 end
6 return Q3 containing the best solutions (the 'decrypted text(s)');
```

Two variants of the PPM modelling system, PPMD and PPMC models have been used in our method. Also two forms of these schemes are examined, one with update exclusions (i.e the standard PPMD or PPMC) (Teahan, 1998) and one without update exclusions. Both of these variants are further investigated with a new variation of the PPM algorithm where a specific high codelength value is assigned to all contexts for which an escape down to an order -1 context has occurred when the symbol being predicted has not already occurred in any higher order context. The idea behind assigning a high codelength value for these order -1 contexts is to penalise these cases during the search as they provide strong evidence of being of lower predictive quality. During the execution of Algorithm 1, these contexts occur frequently at the start since all the characters in the ciphertexts are initialised to the special symbol (full-stop) which is a symbol not found in the 27 character alphabet that is used for the training text. When an order -1 context occurs, the probability can be estimated as $\frac{1}{N}$ where N is the size of the text already processed. Also according to the PPM models that we use, the probability of previously unseen characters such as the special full-stop character does not subsequently change as the ciphertext is being processed. Therefore, we can simply use a fixed codelength value for all the order -1 contexts which is equal to $-\log_2 \frac{1}{N}$. (The size of the training data we use in our experiments is $N = 21,824,832$ so the specific codelength value we assign for order -1 contexts is 24.38.)

To organize and clarify our results, our experiments are divided into different variants as presented in Table 3.2. For the PPM-based variants, both order 4 and order 5 models are used in our experiments as discussed below. Experiments with a full range of variations have been conducted (PPMC, or PPMD, with and without update exclusions, with and without explicit order -1 codelengths; Gzip; and Bzip2). However, for the purposes of this chapter, only the results for the seven variations in the table are shown in order to illustrate either the best performing schemes or to illustrate interesting results for comparison.

Table 3.2: Compression method variants used for the automatic cryptanalysis of simple substitution ciphers.

Name	**Compression method**
Variant A	PPMD without update exclusions
Variant B	PPMD without update exclusions with the same specific codelength value assigned to all order -1 context predictions
Variant C	Standard PPMD (i.e with update exclusions)
Variant D	Standard PPMD with the same specific codelength value assigned to all order -1 context predictions
Variant E	PPMC without update exclusions with the same specific codelength value assigned to all order -1 context predictions
Variant F	Bzip2
Variant G	Gzip

According to the first experimental variant in Table 3.2, Variant A, PPMD5 without update exclusions is applied to compute the codelength values. In Variant B, a new variation of the PPM method is used which is PPMD without update exclusions with specific order -1 codelength values. Both order 4 and order 5 PPMD models are examined. The standard version of the PPMD5 compression method with update exclusions is used for Variant C. The fourth variant, Variant D, is the standard order 5 PPMD model but with specific order -1 codelength values being used instead of the standard PPM order -1 encoding method. Another new version of PPM is used for variant E, PPMC without update exclusion but using the order -1 codelength method. Both order 4 and order 5 PPMC models are examined. For the last two variants, variants F and G, we examine the effectiveness of using the other compression methods Gzip and Bzip2 for computing the codelength metric using the relative entropy calculation as discussed above.

3.5 Experimental Results

In this section, we discuss experimental results for Variants A to G (as detailed in Table 3.2). In our experiments described in this chapter and in the next two chapters, we use the Brown corpus (Francis and Kucera, 1982) and nineteen of the twenty novels used by Irvine (1997) except the novel 'Principles of Computer Speech (book2)' (Witten, 1982), which was unavailable, in order to train our models. These texts were used to train the models in this chapter using 27 character English text. A corpus of 110 different ciphertexts chosen at random from many different resources (including newspapers, magazines and examples from Hart (1994); Lucks (1990)) as testing texts are used, samples of which are listed in Appendix I. The lengths of these ciphers range from 20 characters to almost 300 characters. Table 3.3a and table 3.3b present samples of decryption.

Table 3.3a shows the output sample showing the execution of the algorithm for the ciphertext ' `zjyvgelyzjgqwyzjykoaakbyjgaejvb`' including intermediate results. Compression codelengths with the lowest five results are listed in bits. Table 3.3b presents the ten best solutions as a result of our method of the automatic ciphertext only attack of the simple substitution cipher when using the new version of the PPMD modelling system (labeled as Variant B). In this case, text with the shortest codelength value (the best solution), represents the valid decrypted text. According to the example (in Table 3.3b), the best solution was '`so much sound so little outcome`',

Table 3.3a: Example output.

```
Ciphertext: zjyvgelyzjgqwyzjykoaakbyjgaejvb
Processing the 1st most frequent character 'j';
Buffer length is: 27.
Codelength = 636.168: . ....... ..... ........ ... ..
Codelength = 640.479: .e.......e.....e........e...e..
Codelength = 642.719: .t.......t.....t........t...t..
Codelength = 643.671: .a.......a.....a........a...a..
Codelength = 643.956: .o.......o.....o........o...o..
Processing the 2nd most frequent character 'y';
Buffer length is: 500.
Codelength = 532.282: . t....t. ...t. t......t ... ..
Codelength = 532.948: .e .... .e... .e ...... e...e..
Codelength = 532.951: .t .... .t... .t ...... t...t..
Codelength = 534.603: .s .... .s... .s ...... s...s..
Codelength = 535.227: . s....s. ...s. s......s ... ..
Processing the 3rd most frequent character 'g';
Buffer length is: 500.
Codelength = 467.465: .t .h.. .th.. .t ...... th..t..
Codelength = 470.531: . t.a..t. a..t. t......t a.. ..
Codelength = 470.643: .t .o.. .to.. .t ...... to..t..
Codelength = 471.204: .e .n.. .en.. .e ...... en..e..
Codelength = 471.567: . t.i..t. i..t. t......t i.. ..
Processing the 4th most frequent character 'a';
Buffer length is: 500.
Codelength = 404.402: .t .h.. .th.. .t ..ee.. the.t..
Codelength = 408.380: .ti.h..i.th..i.ti..  ..ith .t..
Codelength = 408.697: .ht.e..t.he..t.ht..  ..the .h..
Codelength = 408.945: .er.d..r.ed..r.er..  ..red .e..
Codelength = 409.339: .t .h.. .th.. .t ..oo.. tho.t..
...
Processing the 8th most frequent character 'b';
Buffer length is: 500.
Codelength = 216.382: hetor..ther..theti.  inter .eon
Codelength = 217.041: e sai..se i..se so.ttons it. an
Codelength = 217.069: so mu.. sou.. so l.ttle out.ome
Codelength = 217.460: t sai..st i..st se.nners in. ar
Codelength = 217.894: to un.. ton.. to d.eeds one.ous
Processing the 9th most frequent character 'e';
Buffer length is: 500.
Codelength = 167.839: so muc. sou.. so l.ttle outcome
Codelength = 172.087: t shad.st a..st si.nnies and he
Codelength = 173.728: t spad.st a..st se.nners and pr
Codelength = 174.211: he san. hea.. he o.rrot earnest
Codelength = 176.220: he rat. hea.. he i.ssin eastern
Processing the 10th most frequent character 'q';
Buffer length is: 500.
Codelength = 145.242: so muc. soun. so l.ttle outcome
Codelength = 147.082: so muc. sour. so l.ttle outcome
Codelength = 147.961: so muc. soug. so l.ttle outcome
Codelength = 150.597: so muc. soup. so l.ttle outcome
Codelength = 151.714: t shad.st al.st si.nnies and he
...
Processing the 13th most frequent character 'w';
Buffer length is: 500.
Codelength = 63.884: so much sound so little outcome
Codelength = 72.105: so much sourd so little outcome
Codelength = 73.876: so much soung so little outcome
Codelength = 75.474: so muck sound so little outcome
Codelength = 75.519: so much sound so lyttle outcome
```

Table 3.3b: Example output (ten best solutions).

Ten best solutions	
63.884	`so much sound so little outcome`
72.105	`so much sourd so little outcome`
73.876	`so much soung so little outcome`
75.474	`so muck sound so little outcome`
75.519	`so much sound so lyttle outcome`
76.426	`so much sound so lattle outcome`
76.677	`so mucy sound so little outcome`
79.165	`so muck sough so little outcome`
79.350	`so much sourg so little outcome`
79.896	`so much souvy so little outcome`

which has the shortest compression codelength value 63.884 and is the valid decrypt.

To encrypt the plaintext (original text), a random key is generated for each run. Afterwards, the attack is performed on the cryptogram. Various ciphertexts with different lengths (even very short) have been examined in our experiments. We experimented with 110 different ciphertexts. In order to measure the success and the accuracy of our automatic cryptanalysis algorithms, alphabetic substitution errors (mapping errors) are counted. The results of our experiments showed that only when using the new PPM variants (Variants B and E), as a method of calculating the codelength values were almost all the ciphertexts decrypted successfully. In contrast, the other PPM variants produced a significantly greater number of errors. The same was repeated when using the Gzip and Bzip2 algorithms in the last two variants (F and G). Example output produced by the different variants is shown in Table 3.4.

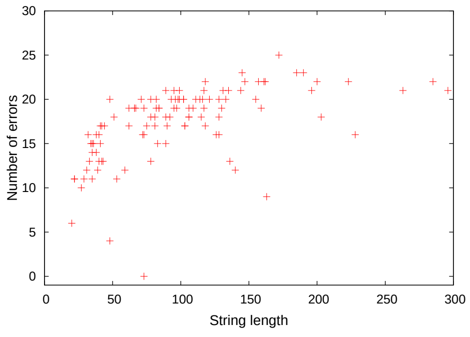
For variant A, Figure 3.1a presents the number of errors for each testing cryptogram as a result of our automatic cryptanalysis method using PPMD without update exclusions. Clearly, we can see that the number of errors for most tested cryptograms are high. In this case, just one cryptogram is solved with no errors, and only four examples are found to have ten errors

Table 3.4: Sample of solved cryptograms by different variants.

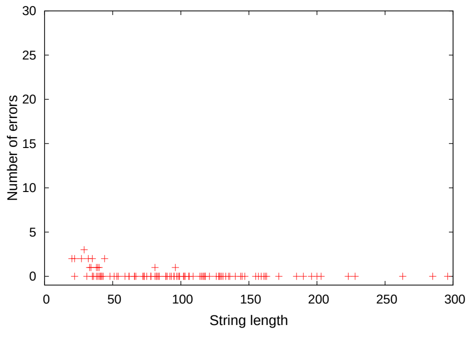
Variants	**Number of errors**	**Example decrypted message**
Variant A	17	ladylamandems dejaegonzalpsteyemainerosecine scheynero-sle domickepoleontre oetonw
Variant B	0	retirement must be wonderful i mean you can suck in your stomach for only so long
Variant C	20	nod nineosiukosqnsprean tuysdsincesbrushceskuhlsdesbru skorichmstr sreybskrsyrex
Variant D	19	spaxsprpia roma hp cviupsloe x rpyi tvo nyi monk xi tvos mavrynw lvs viet mv evid
Variant E	2	retirement must be wonderful i mean you can such in your stomack for only so long
Variant F	21	myr myaywruatiruzyugvwhymotbu uaydwuxvtuedwuitenu wuxvtmuirvadekuovmuvwbxuivubvwc
Variant G	11	detadement mrst he pongedbrl a mein for cin srck an ford stomicy bod onlf so lonj

or less. The results show that over 75% of the decrypted cryptograms have more than ten errors and 20% have greater than twenty errors.

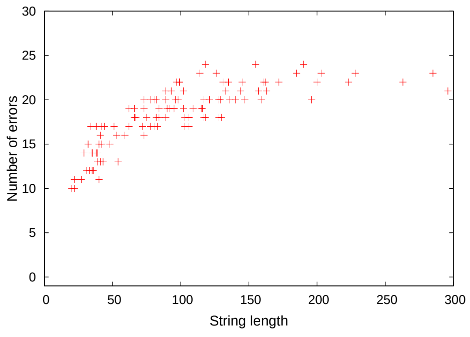
For variant B, both order 4 and order 5 PPMD models are examined. For the order 4 model, the results show that 81% of the ciphertexts are correctly solved with no errors (that is, the best solution with minimum codelength value is the correct solution). About 19% of the examples are decrypted with just three errors or less. For the order 5 model, the results show better performance with over 87% of the cryptograms correctly solved without any errors. Also 12% of the ciphers are decrypted with just one or two errors, and only one example had three errors as shown in Figure 3.1b. Moreover, in almost all these examples, the correct solution can also be found within the ten best solutions. It is clear that the number of errors for this variant is much lower than other variants.



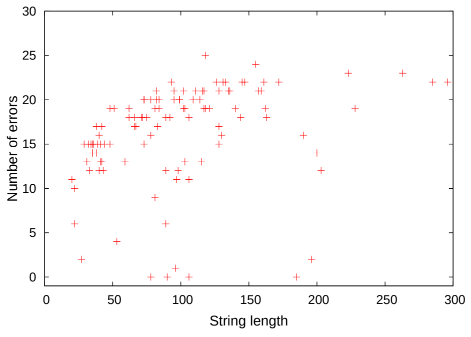
(a) Errors produced from variant A.



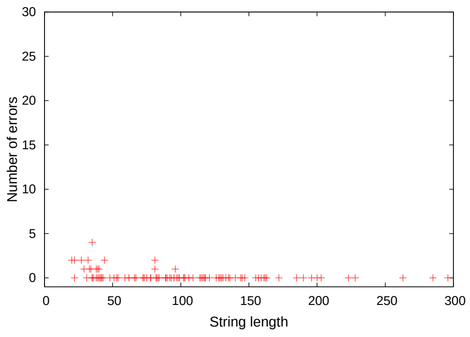
(b) Errors produced from variant B.



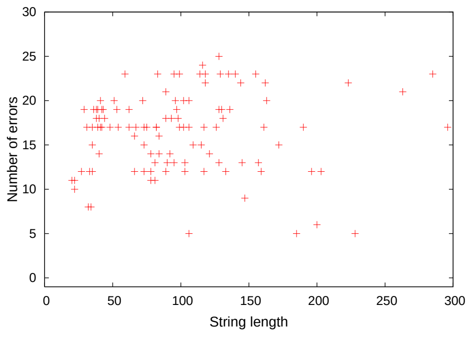
(c) Errors produced from variant C.



(d) Errors produced from variant D.



(e) Errors produced from variant E.



(f) Errors produced from variant G.

Figure 3.1: Errors produced from different variants

Figure 3.1c illustrates the number of errors for each cryptogram for variant C. We can see that almost all the decryption errors are more than ten, with just two examples being solved with ten errors. Over 71% of the cryptograms are decrypted with greater than ten errors, and over 26% of the examples have more than twenty errors. None of the examples produced no errors.

In variant D, the results show that most of the errors are greater than 10 with just four of the ciphertexts solved without errors. Over 58% of the decrypted cryptograms have ten errors or more, and approximately 32% have twenty errors or more with just 6% having less than ten errors.

Variant E produces slightly worse results than variant B, with 80% of examples having been successfully solved without any errors and 20% decrypted with four errors or less when using the order 4 model. The order 5 PPMC model produces slightly better results. When we examine only the best solution, 86% of examples are successfully decrypted without any errors, and 13% solved with one or two errors, and one of the decrypted cryptogram having four errors. When we examine the ten best solutions, almost all the examples have no errors. Figure 3.1e presents the results of variant E using the order 5 model.

We also experimented with using our relative entropy calculation method using Bzip2 for Variant F. But due to the block-sorting nature of the Burrows-Wheeler algorithm, the calculation of some of the relative entropy code-lengths ended up being negative. (It is not clear why this is so but it maybe due to the additional text aiding the run-length encoding of the blocks). Thus these results could not provide us with a complete picture with regards to the average number of errors. However, none of the positive results for Variant F did show any success, with a quite high number of errors.

The number of errors produced form variant G is shown in Figure 3.1f. It is clear that the number of errors for each decrypted ciphertext is much higher, with most of the errors being greater than 15. Also none of the cryptograms offered no errors and just seven cryptograms were decrypted with the number of errors being less than 10.

Results regarding the average number of automatic cryptanalysis errors for the 110 cryptograms we tested with each of the variants are presented in Tables 3.5a and 3.5b. Table 3.5a presents the average number of errors when just examining the best solution, and Table 3.5b shows the average errors for the ten best solutions. Clearly, the best performing model overall is PPMD5 without update exclusions using the order -1 correction model (Variant B). However, Variant E, which used PPMC5 without update exclusions along with order -1 correction, presents excellent results as well. On the other hand, the other Variants A, C and D produce poor results. Interestingly, the PPM without update exclusions method, which typically shows slightly worse performance at the compression task, shows better performance at decryption here.

Table 3.5a: Average number of errors for each different variant when examining the best solution.

Variants	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Average errors	17.37	0.20	18.29	16.21	0.22	–	16.59

The average number of errors produced for Variant G (using Gzip2) is presented in the last column in the table. The results show that the Gzip compression scheme is not an effective way of recognising the valid decryptions as it also results in a high number of errors. In addition, the time that is needed to break the ciphers (by using Gzip and Bzip2) using the relative entropy calculation is considerably longer (as it involves repeatedly compressing the training text), and thus also makes the use of these methods

Table 3.5b: Average number of errors for each different variant when examining the ten best solutions.

Variants	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Average errors	17.45	1.38	18.31	16.28	1.40	–	16.83

impractical.

Like other cryptanalysis approaches, very short cryptograms can often not always be solved correctly, even with a better model of English as Irvine claimed in his thesis (Irvine, 1997). This is because these cryptograms are inherently ambiguous as a simple substitution unicity distance is about 26, according to this equation:

$$U = \frac{H(k)}{R} = \frac{\log 27!}{\log 27 - 1.2} = 26.2$$

where U represents the unicity distance, $H(k)$ denotes the entropy of the key space and R is the plaintext redundancy in bits per character (Irvine, 1997).

For an order 4 model, the unicity distance is equal to 33.2, since the entropy of this model is $H_4 = 1.953$. For an order 5 model,

$$U = \frac{\log 27!}{\log 27 - H_5} = \frac{\log 27!}{\log 27 - 1.822} = 31.8.$$

So, we can not expect to correctly decrypt cryptograms shorter than this number of symbols. However, in our approach (Variant B), many different ciphertexts with short lengths (ranging from 20 to 40) have been tried, and in almost all cases the right solution (without any errors) was found. Table 3.6 lists some examples of different cryptograms with short lengths and the successfully decrypted text.

Table 3.6: Examples of decryption of short cryptograms (length from 20 to 40 characters; the compression codelength h and compression codelength ratio $H = h/n$, where n is the number of characters in the text, are shown in the first two columns).

h	H	**Ciphertext**	**Successfully decrypted text**
30.77	1.538	fvwdwradbsdfvwdobija	the end of the world
37.76	1.716	yniseiyre sgosynisbrvy	the return of the suit
64.71	2.231	fbapipuymswykpdbpubjypvumttyt	how i learned to love glasses
68.50	2.140	cgjgulg flrmuglfv clomxli clwhyf	never eat more than you can lift
74.84	2.138	vlmvhhvwwnlemtnwqljqmrlmekl mjrlasrw	an appalling silence on gun control
77.12	2.142	larrpmvcrnjil jm txm mc sspmtaw mpa r	merry christmas and a happy new year
72.72	1.818	igecq crgirwqcrkbcnfrnbfrwhr wq xcbinfgwq	silence is one great art of conversation

The average execution time that is needed to determine the correct solutions of the different ciphertexts that were experimented with is presented in Table 3.7. The time which is required to automatically break each ciphertext is based on the execution of the PPMD model without update executions and with specific order -1 codelength value (Variant B). The average elapsed time in seconds for each cryptogram is computed by running the program ten times on a Dell Inc.-Inspiron 5537 laptop computer (Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz) and then calculating the average. The results show that our method only requires a few seconds on average to decrypt the ciphertexts, and usually the solution is found in less than six seconds of CPU time. This is compared with an average execution time of approximately 76 minutes for a ciphertext of 64 letters in the research by Ravi and Knight (2008), and with a reported decryption time of two and a half minutes for a ciphertext of the same length in the paper by Nuhn et al. (2013). However, these are not directly comparable because different processors are being used in these experiments using different set-ups.

Table 3.7: Average time needed to automatically cryptanalyse different simple substitution ciphertexts.

Cipher Length	**20**	**50**	**150**	**300**
Time *(Sec)*	2.22	2.61	3.26	5.57

3.5.1 Experiments with Different Buffer Sizes

Our search method requires maintaining a current sorted list of the best solutions using a buffer of fixed size. In order to determine which is the best or the most appropriate buffer size for obtaining the best results, we performed four further experiments using different buffer sizes: 100, 200, 500 and 1000. Table 3.8a and 3.8b present results regarding the average number of errors for the 110 ciphertexts when using the different buffer sizes.

According to these tables, it is clear that buffer sizes of 500 and 1000

Table 3.8a: Average number of errors when using different buffer sizes when examining just the best solution.

Array size	**100**	**200**	**500**	**1000**
Average errors	1.35	0.70	0.20	0.20

Table 3.8b: Average number of errors when using different buffer sizes when examining the ten best solutions.

Array size	**100**	**200**	**500**	**1000**
Average errors	2.33	1.70	1.38	1.38

produced the smallest average number of errors. In contrast, using a buffer size of 100 resulted in a greater number of errors (1.35 compared to 0.20). However, a trade-off in favour of a smaller buffer size is that it uses up less memory and execution speed is faster. The program has not been optimised for memory usage and execution speed; however, we have noticed that the execution time doubled with the size of the buffer.

3.5.2 Improving Results Using a Word-based PPM Compression Method

This section discusses the experimental results obtained when using a further word-based model for the automatic cryptanalysis. As word-based schemes ($W|W$ as described in Table 3.1) for the English text outperform character-based ones in terms of compression rate, the order 1 word-based model is used here to examine the effect of applying this model in a secondary post-processing stage (Algorithm 2 as above) on the output from Algorithm 1 to see if this results in better cryptanalysis. This model is used to re-order the solutions produced from Algorithm 1 according to the codelength value calculated for each solution using the same model. The smaller the codelength value, the more closely the solution represents the valid decrypt.

Some sample output is shown in Table 3.9 for the ciphertext ‘cgjgulg

Table 3.9: The ten best character-based solutions compared to the ten best word solutions for the ciphertext '`cgjgulg flrmuglfv clomxli clwhyf`'.

Ten best character solutions:		**Ten best word solutions:**	
65.994	`never eat more than you can dist`	64.041	`never eat more than you can lift`
66.439	`never eat more than you can spit`	68.016	`never eat more than god pan just`
68.030	`never eat more than you can sixt`	68.219	`never eat more than you can list`
68.388	`never eat more than you can lift`	69.054	`never eat more than you can gift`
68.453	`never eat more than you can gift`	69.441	`never eat more than god can just`
68.528	`never eat bore than you can dist`	69.804	`never eat more than you can fist`
68.745	`never eat more than you can list`	70.263	`never eat more than you can spit`
68.974	`never eat bore than you can spit`	70.456	`never eat more than you can wilt`
70.333	`never eat wore than you can dist`	70.678	`never eat more than you can jist`
70.565	`never eat bore than you can sixt`	70.804	`never eat more than you can gilt`

flrmuglfv clomxli clwhyf' which compares the ten best character-based solutions found by Algorithm 1 with the ten best word-based solutions found by Algorithm 2. The sample shows that the correct solution was found by the word-based method (with the semantically correct last word 'lift'), but in comparison this was ranked in fourth place using the character-based method. Interestingly, both methods have found similar solutions except for the third and last words which in most cases are correctly spelt although semantically incorrect.

This technique was tried only for variants found to be the best performing, Variants B and E. For variant B, five examples out of fourteen (which had been found by the character-based method to have three errors or less), were successfully solved with no errors when this secondary word-based method was applied. For variant E, six examples out of fifteen (which had been found to have three errors or less), are also solved with no errors using the same method.

After applying this word-based method, 92% of cryptograms were now solved without any errors, with an improvement of 5% and 6% for both variants B and E over when just using the order 5 character-based model. Furthermore, all the cryptograms of length longer than 44 are successfully solved without any errors. Table 3.10 shows how the word-based approach improves the average number of errors for the best solutions.

Table 3.10: Average number of errors when examining the best solution.

Variants	**Order 4 character model**	**Order 5 character model**	**Word-based ($W|W$) model**
Variant B	0.29	0.20	0.13
Variant E	0.31	0.22	0.13

Table 3.11 presents the summary of results when using our new method Variant B (the best performing method) on the ciphertexts that we experimented with. Overall, the results show that we are able to attain a very high success rate, with about 92% of cryptograms being correctly solved with no errors and 100% being decrypted with just three errors or less.

Table 3.11: Summary of results for Variant B.

Errors	Order 4 character model		Order 5 character model		Order 4/5 character & word-based models	
	No. of ciphertexts	Cumulative percentage (%)	No. of ciphertexts	Cumulative percentage (%)	No. of ciphertexts	Cumulative percentage (%)
0	89	80.91	96	87.27	101	91.82
≤ 1	101	91.82	105	95.45	106	96.36
≤ 2	108	98.18	109	99.09	109	99.09
≤ 3	110	100.00	110	100.00	110	100.00

3.6 Conclusions

In this chapter, a new method for the plaintext recognition and automated cryptanalysis of substitution ciphers in a ciphertext only attack has been described. An efficient use of the compression-based approach for cryptanalysis has been demonstrated. The fundamental idea behind our approach relies on using a compression method as an accurate way of measuring information in the text. Results on 110 cryptograms ranging from 20 to 300 characters have shown a very high success rate with approximately 92% of the cryptograms correctly decrypted without any errors and 100% with just three errors or less (which were due to alphabetic mapping errors). This efficient method works well on even very short ciphertexts and eliminates any need for human intervention. This is a significant improvement over the earlier work done by Irvine (1997) which used a different version of PPM and a different search algorithm.

Three main compression methods have been investigated: prediction by partial matching (or PPM), Gzip and Bzip2. Various character-based PPM variants were investigated as well, in order to ascertain the most effective scheme when applied to the problem of automatically breaking simple substitution ciphers. The following variants of PPM were used: PPMD and PPMC, with further variations such as the use or not of update exclusions, a technique found to improve compression but which we have found leads to better decryption if it is removed. Both of these variants were further investigated with a new variation of the PPM algorithm where a specific codelength value is assigned when encoding all order -1 contexts. The experimental results showed that this new combination, PPM without update exclusions using specific order -1 codelength values, noticeably outperforms other compression schemes including Gzip and Bzip2. We have also applied a word-based PPM model as a post-processing stage which led to further improved results.

Chapter 4

Designing and Evaluating a New Automatic Cryptanalysis of Transposition Ciphers Using Compression

4.1 Introduction

In this chapter, we propose a novel compression-based approach applied to the problem of automatically decrypting transposition ciphers. This seeks to address research questions 1 and 2 that were detailed in Section 1.2, and fulfils objectives 2 and 3 in Section 1.3. In essence, we investigate how to devise better solutions to the plaintext recognition problem by using transposition ciphers as a test bed. Furthermore, we propose further methods also based on using compression to automatically insert spaces back into the decrypted texts in order to achieve readability (as we perform our experiments on English alphabetic characters). This helps to address both research question 4 and objective 4 that were listed in Chapter 1. Specifically, the use of PPM and Gzip compression methods are examined in this chapter.

A preliminary form of the work in this chapter was published in the Proceedings of the 15th International Conference on Cryptology And Network Security (CANS2016), Springer International Publishing (Al-Kazaz et al., 2016).

This chapter is structured as follows. Section 4.2 gives a brief description of the previous research into the cryptanalysis of transposition ciphers. In Section 4.3, we illustrate the motivation for the use of our compression-based approach as a method of tackling the plaintext recognition problem and the word segmentation problem. Section 4.4 presents the new computerized method and the pseudo-code we developed for this work. Our experimental results are discussed in section 4.5. Section 4.6 summaries the results.

4.2 Related Work

Various cryptanalysis methods have been used to break transposition ciphers, starting with traditional attacks such as exhaustive search and anagramming, and then leading to genetic algorithm based methods. Anagramming is a well-known traditional cryptanalysis method. It is the method of repositioning disarranged letters into their correct and original positions (Sinkov, 1966; Seberry and Pieprzyk, 1989). Although, the traditional attacks are more successful and easy to implement, but automating these types of attack is not an easy issue. It requires an experienced and trained cryptanalyst. Mathematical techniques have been used in these attacks but the main role tends to be on the human expert. The final decision is made by the human cryptanalyst with regards to which algorithm is used in attack.

Many researchers have been interested in developing and automating cryptanalysis against transposition ciphers. One of the earliest papers was published by Matthews (1993). He presented an attack on transposition ciphers using a genetic algorithm. The fitness function was based on the frequency of the common English digrams and trigrams that appear in the deciphered text. A test text with a length of 181 characters was selected, and three targeted keylengths (7, 9 and 11) were experimented with. This attack was successful at key size of 7, with no successes at key length of 11.

Clark (1994) published three algorithms that used simulated annealing, genetic algorithm and tabu search in the cryptanalysis of transposition ciphers. The fitness function used also depended on the frequencies of digrams and trigrams. By using a genetic algorithm, the success rates of block sizes of 4 and 6 ranged from 5 to 91%. Tabu search was faster than the other algorithms while simulated annealing was the slowest but with a high performance of solving ciphertexts especially with large periods. It was able to correctly recover 26 of the key elements, for a transposition cryptosystem of period 30 with 1000 ciphertext characters. For periods less than 15, each of the algorithms could effectively recover the key (Clark, 1998). Dimovski and Gligoroski (2003) came to similar conclusions presented in Clark's publication. For a key of length 15, a ciphertext of at least 800 letters are required to recover 12 key elements out of 15. For a key of length 30, at least 1000 ciphertext letters are required to recover 25 key elements out of 30. The fitness function that was used in their paper was based on bigram statistics due to the expensive task of calculating trigram statistics.

Toemeh and Arumugam (2007) used a genetic algorithm and a slightly modified list of the most common trigrams than were used in Clark paper to break transposition ciphers. Three additional trigrams were included with the Clark table. The recovered key for a ciphertext of 1000 letters was 15 out of 15 key elements.

Genetic algorithms for the cryptanalysis of transposition ciphers were published by Grundlingh and Van Vuuren (2003); Bergmann et al. (2008). The fitness function they used in their research was based on the discrepancies between the expected number of occurrences of a digram in a natural language text (per N characters), and the observed count of this digram in a ciphertext of length N. Grundlingh and Van Vuuren (2003) concluded that genetic algorithmic attacks were not effective against columnar transposition ciphers since this cipher is more robust than substitution ciphers. This attack was only successful at a key size of 7 with a ciphertext of 2519 characters (Grundlingh and Van Vuuren, 2003). A transposition cipher (Bergmann et al., 2008) with a key size of up to 12 and 500 characters in length was able to be deciphered correctly using the proposed algorithm.

Giddy and Safavi-Naini (1994) used a simulated annealing approach and bigrams based cost function for the automatic decryption of transposition ciphers. A success rate of at least 80% was obtained in the following cases: for a key of length 15 with a ciphertext of 255 characters, key of lengths 20, 25 with a ciphertext of 500 characters. The algorithm was not be able to correctly decrypt short ciphers of 100 characters or less, which they noted is the supposed behaviour of all cryptanalysis schemes. Ciphertexts that have dummy characters added to them were decrypted poorly as well. (In the case of a short block at the end, a dummy character, sometimes an ‘X’, is used to fill in the blank cells).

The use of genetic algorithms was specifically explored by Delman (2004). Different key lengths ranging from 2 to 30 with different ciphertext lengths were examined. None of these algorithms were able to correctly recover all the plaintext and achieve full success. Delman (2004) concluded that the genetic algorithm-based approach did not deserve further effort and further investigation in traditional cryptanalysis techniques was warranted rather than for genetic algorithms.

Other local search metaheuristics for the automatic cryptanalysis of transposition ciphers were proposed by a number researchers (Russell et al., 2003; Chen and Rosenthal, 2012; Wulandari et al., 2015; Jassim, 2017). Russell et al. (2003) used a dictionary to recognise the plaintext and bigrams to indicate adjacent columns. This attack was able to decipher a ciphertext of 300 letters with a key of length 15, a ciphertext of 400 letters with a key of length 20 and a ciphertext of 625 letters with a key of length 30. Chen and Rosenthal (2012) used bigram statistics as the basis for calculating the score function. This method showed a very high accuracy rate with a key of length 20 and 2000 ciphertext characters. To solve a key of length 30 with 80% probability of success, a ciphertext of 2000 letters were required. The fitness function used in the paper published by Wulandari et al. (2015) was based on bigram and trigram statistics. Texts with different sizes (665, 822, 980, 2316, 3812) were tested. This algorithm was able to decrypt the ciphertexts correctly with key lengths up to 9; with key length of 10, it was able to find half of the correct answers.

An enhanced hill climbing algorithm with two phases for the automatic cryptanalysis of the columnar transposition cipher was proposed by Lasry et al. (2016). It implemented specialized adaptive scoring, using two new developed scoring methods: the adjacency score and the alignment score. The calculation of these scores was based on using bigram statistics. For shorter keys, an improved implementation of the scoring methods using trigram statistics was performed. These methods were used in the first phase in order to achieve better resilience to errors. Quadgram statistics, which have a very good selectivity, was used as a basis for calculating the score function in the second phase. This work addressed much longer keys, for example, it was able to recover key with length of 30 elements with 180 ciphertext letters.

Irvine (1997) has been the only researcher to date to have used a compression algorithm to break a cipher system, substitution ciphers. However, a similar approach has yet to have been applied to other ciphers systems, including transposition ciphers.

In this chapter, we propose a novel compression-based approach for the automatic recognition of the plaintext of transposition ciphers with a 100% success rate. We use different key lengths (ranging from 2 to 12) and different ciphertext lengths, even very short messages with only 12 characters while the shortest messages used in most of the previous research were not less than 100 characters. In this chapter, we present both a method for automatically decrypting transposition ciphertexts and then automatically achieving readability subsequently. This automatically inserts spaces into the decrypted text, while most of the previous works did not address or refer to this fundamental aspect of the cryptanalysis.

4.3 Compression as a Cryptanalysis Method, using PPM Compression Codelength Metric and the Gzip Compression Method

Our approach adopts a similar method found successful for simple substitution ciphers as described in the previous chapter. One of the critical factors for a successful attack, and often the most critical one, is the evaluation function (scoring function). In our approach here, compression schemes are used as effective scoring methods and calculate the compression codelength value for each possible permutation in order to evaluate the quality of each of them. Achieving readability is the second step of our approach which in turn also depends on compression methods. We use PPM with the Viterbi algorithm for the word segmentation problem and this is explained in the next subsection. (Another new method that has been developed again using PPM for segmenting words is described in detail in the next section).

4.3.1 Word Segmentation Using the Viterbi Algorithm

Word segmentation is the process of determining the smallest unit (word) in a meaningful context (Alhawiti, 2014). It is an important task for some natural language processing applications, such as speech recognition.

Character-based PPM models with the use of the Viterbi algorithm (Viterbi, 1967) has achieved a high accuracy rate for the word segmentation of English text (Teahan, 1998). This model works by searching through all alternative segmentations of the text (by inserting spaces after each letter for each possible segmentation). For each letter, there are two possible segmentations: the letter itself and the letter followed by a space. The segmentation is chosen by selecting the one that has the best encoding performance as determined by the PPM compression model.

There are 2^n possible sequences to be searched, where n denotes the sequence length. However, much of the search space is eliminated using the Viterbi algorithm through the substantial pruning of underperforming sequences. How this works is shown in Figure 4.1. For each similar context, after the processing of each letter, only one path should remain using the PPM model. For example, the substring "tob" would have eight search possibilities and the most probable segmentations would be identified by the PPM model as illustrated in Figure 4.1. An order 2 PPM model is used in the example and the poorer performing paths which have the same order 2 context are discarded. The best encoding sequence for the three order 2 contexts "ob, _b and b_" are 16.3, 20.9 and 21.3 bits respectively. These sequences are the highest probability paths that are kept and subsequently expanded with the remaining five poorer performing paths being discarded (as shown in the figure), as these paths can not perform better than the other paths.

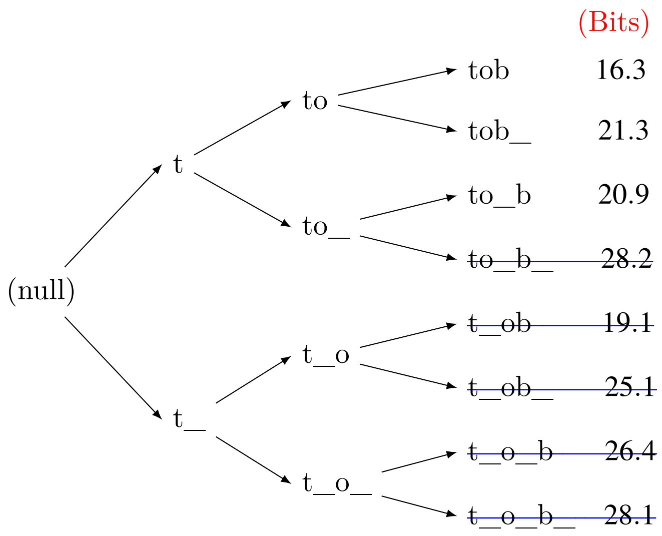


Figure 4.1: Segmentation search tree (Teahan, 2018).

4.4 The New Method

In this section, we give a full description of the new approach for the automatic cryptanalysis of transposition ciphertexts. As before, the basic idea of our approach depends on using a compression model as a method of computing the 'codelength' of each possible permutation. This compression model and the codelenghth metric represent the evaluation function that can be relied on it to automatically rank alternative permutations and recover correct messages. In our method, the PPMD, PPMC and Gzip compression methods are used in the experiment.

Our new approach consists of two essential phases. The main idea of the first phase (Phase I) depends on trying to break a ciphertext automatically using a transposition of specified size by exhaustively computing all possible transpositions. The second phase (Phase II) focuses on inserting spaces automatically (segmenting words) into the decrypted message which is outputted from the first phase (since we remove spaces from the ciphertext at the beginning of Phase I, as is traditional).

The pseudo code for the first phase of our method is presented in Algorithm 3. The first step in this algorithm focuses on removing all the non-alphabetical characters (including spaces) from the ciphertext (see line 1). At this stage, the text comprises just 26 alphabetic English characters. The algorithm then starts to try all possible key sizes and for each key size a permutation is performed over each cryptogram block trying to get a permutation with a smaller codelength value which represents the correct solution (lines 4 to 13). Each cryptogram is divided into blocks according to the key size (lines 5 and 6). Then, a permutation is performed over each cryptogram (line 7). For each possible permutation, a codelength value is calculated (lines 8 to 11). The text is compressed using PPM with an order 5 character-based model of English trained on nineteen novels and the Brown corpus as this has been found the most effective (line 9). The resultant size of the compressed text is used to accept or reject the particular permutation.

Gzip is also used for calculating the codelengths. Permutations with smaller codelengths are kept in the priority queue as shown in line 10. The hope is that the cryptogram that has the smallest codelength value will represent the valid decrypted message. We have found in fact that the smaller the codelength, the more closely the cryptogram resembles the model.

Algorithm 3: Pseudo-code of the main decryption phase (Phase I) for transposition ciphers.

```
   Input  : ciphertext
   Output: decrypted text
 1 remove all non-alphabet characters and spaces from the ciphertext;
 2 maximum size of Q (priority queue) ← 3;
 3 maximum key-size of transposition ← 12;
 4 foreach key-size do
 5     if ciphertext-length mod Key-size = 0 then
 6         divide the ciphertext into blocks according to key-size;
 7         perform a permutation over each ciphertext blocks;
 8         foreach possible permutation do
 9             calculate codelength value using PPM or Gzip compression model;
10             store a permutation with smaller codelength value in Q;
11         end
12     end
13 end
14 return the priority queue 'Q' (the 'decrypted text');
```

As the output of the previous phase are texts without any spaces, the second phase focuses on segmenting words. Two alternative ways have been investigated in this phase. The first method which is called “Phase II-A”, examines all further decrypted message possibilities when a space is inserted after each character. In this new method, possibilities with smaller codelength values (best performing possibilities) are kept in a priority queue, and those which showed poor codelength values are pruned (see Algorithm 4). For each of the text produced as output from Algorithm 3 (lines 3 to 19), it repeatedly tries to improve the codelength of each solution (lines 5 to 17) by adding a space after each character. For each character in the text, another new solution is created after a single space character is added (lines 8 to 12). Compression codelength then is applied to rank the solutions. PPM and Gzip compression models are also used again in this method.

Referring to Algorithm 4 in more detail, a number of queues are used during the processing. Queue ‘Q’ is the output produced from Algorithm 3. Q1 contains text permutations of possible segmentations (i.e. alternative space insertions) for a text being processed from Q. Q3 is being dynamically built through an iterative improvement process (lines 3 to 19) and is used to produce the output from the algorithm. Alternative segmentations are searched on lines 7 to 13, and these are placed in a temporary queue Q2. If there is an improvement in codelength value, then Q2 is swapped for Q1 (line 15). This iterative improvement process is repeated (lines 5 to 16) until there is no further improvement. Then the first text in Q1 is added to Q3 which is returned by the algorithm (line 20).

The second method, which is called Phase II-B, uses the Viterbi algorithm to find the best possible segmentation (see Algorithm 5). For each of the text produced as output from Algorithm 3, the Viterbi algorithm is used to search for the best segmentation sequence and this then is stored in Q1 (lines 2 to 5) which is used to return the result (line 6). PPM compression model is used again here.

In our method, we have used two variants of the PPMD and PPMC models, one without update exclusions (Teahan, 1998) and the standard PPMD. This was done to investigate which is the most effective model when applied to the problem of the automatic cryptanalysis of transposition ciphers.

In order to clarify and organize our experiments and results, we divide our different experiments into different variants with a specified label as shown in the Table 4.1.

According to Table 4.1, the first variant is called Variant A. In this variant, PPMD and PPMC without update exclusions are used to calculate the compression codelength values. This is used for the main decryption

Algorithm 4: Pseudo code of the second phase (Phase II-A) for transposition ciphers.

```
Input  : the priority queue ‘Q’ from Phase I
Output: segmented decrypted text
1  maximum size of Q1, Q2 (priority queues) ← 5;
2  maximum size of Q3 (priority queue) ← 1;
3  foreach text in Q do
4      Q1 ← text;
5      repeat
6          Q2 ← empty;
7          foreach text in Q1 do
8              foreach character in a text do
9                  create a new text with a single space added;
10                 calculate codelength value for the new text using PPM or Gzip
                   compression model;
11                 store new message that have a smaller codelength value in Q2;
12             end
13         end
14         if there is any improvement in the codelength value then
15             Q1 ← Q2;
16         end
17     until there is no improvement in the codelength value;
18     add the first text in Q1 to Q3;
19 end
20 return the best segmented decrypted text from Q3;
```

Algorithm 5: Pseudo code of Phase II-B for transposition ciphers.

```
Input  : the priority queue ‘Q’ from Phase I
Output: segmented decrypted text
1  maximum size of Q1 (priority queue) ← 1;
2  foreach text in Q do
3      use the Viterbi algorithm to search for the best segmentation sequences;
4      store the text that have the best segmentation which present in Q1;
5  end
6  return the best segmented decrypted text from Q1;
```

Table 4.1: Variants used in our experiments

Variants	**Phase I**	**Phase II-A**	**Phase II-B**
Variant **A**	PPM with no update exclusions	PPM with no update exclusions	
Variant **B**	PPM with no update exclusions		PPM with no update exclusions
Variant **C**	PPM	PPM	
Variant **D**	PPM		PPM
Variant **G**	Gzip	Gzip	

phase—Phase I and for Phase II-A as well. All cryptograms can be solved using an order 5 model. In the second variant, Variant B, PPMD5 and PPMC5 without update exclusions are used in both phases, Phase I and Phase II-B. The Viterbi algorithm is used in the second phase.

Different versions of PPMD and PPMC compression models are used in the third variant, which is named "Variant C". The standard PPMD5 and PPMC5 (with update exclusion) are used as the method for calculating the codelength values for both phases (Phase I and Phase II-A). Variant D uses the standard order 5 PPMD and PPMC, as well in the calculation of the codelength values. For the second phase, Phase II-B, these compression models are also used as a basis for segmenting the words.

For variant G, we examine the effectiveness of another type of compression method which is the Gzip compression system. The Gzip algorithm is used in the main decryption phase and for the second phase "Phase II-A", as the basis for computing the codelength metric.

4.5 Experimental Results

In our method, the order 5 PPMD and PPMC models were trained on nineteen novels and the Brown corpus using 26 and 27 character (including space) English text. In our experiments, we use a corpus of 90 cryptograms (samples of which are in Appendix II) with different lengths from different resources as testing texts. The lengths of the ciphertexts that have been examined in our experiments range from 12 letters to over 600 letters. Table 4.2 presents a sample of decryption.

Table 4.2: Output sample from the different phases for the ciphertext 'prcy rotg ypah oedm'. (Compression codelengths are listed in bits with the lowest 5 results presented for Phase-II-A.)

Phase I	**Phase II-A**	**Phase II-B**
53.73 cryptographydemo	42.85 cryptography demo	cryptography demo
	50.94 cryptographyde mo	
	59.41 cryptographyd emo	
	59.68 cryptograph ydemo	
	67.64 c ryptographydemo	

A random key is generated to encipher the original text (plaintext) for each run. After that, the attack is performed on the ciphertext. Different key sizes (period or permutation size) and different ciphertexts with different lengths have been experimented in our method. The results of the first phase—Phase I, by using the PPM method, showed that all the valid decryptions were recognised and all the ciphertexts were able to be decrypted successfully with no errors. In our method for all the different variants, except Variant G, we were able to achieve a success rate of 100%. We have used different key size (block sizes) from two to twelve. We experimented with 90 different ciphertexts with different lengths (including very short) and all can be solved correctly. In contrast, by using the Gzip algorithm in the last variant (G), we were only able to achieve a success rate of 94% as result of Phase I.

For each variant, we have performed two types of experiments (except for Variant G). Since in Phase I we deal with texts without any spaces included, our first experiment is done by using PPMD and PPMC models after being trained on 26 English characters (instead of 27) as the basis for calculating codelengths. In the other experiment, PPMD and PPMC compression models trained on 27 character English texts were used. As stated, the output result from these two experiments is the same achieving a 100% success rate.

For the second phase, we used the Levenshtein distance as a metric for measuring the differences between the plaintext and the decrypted text with spaces. Levenshtein distance is a commonly used string metric for counting the differences between two strings (such as insertions, deletions or substitution) (Levenshtein, 1966). In our approach, in almost all cases the correct (readable) solution was found. The next table (Table 4.3) provides example output (with spaces) produced by the different variants.

Table 4.3: Example of solved cryptograms with spaces by different variants.

Variants	**Number of errors**	**Decrypted message (with spaces)**
Variant A	2	an excuse is worse and more terrible than a lief or an excuse is a lie guarded
Variant B	0	an excuse is worse and more terrible than a lie for an excuse is a lie guarded
Variant C	3	anexcuse is worse and more terrible than a lief or an excuse is a lie guarded
Variant D	1	anexcuse is worse and more terrible than a lie for an excuse is a lie guarded
Variant G	14	anexcuseisworse andmoreterrible thanalieforanexcuseisalie-guarded

Figure 4.2 illustrates segmentation errors produced from the different variants when the PPMD models are used. For variant A, Figure 4.2a shows the number of errors for each testing text as a result of the second phase. Clearly, we can see that most of the space insertion errors are less than two. The results show that 50% of texts have correctly inserted spaces with no

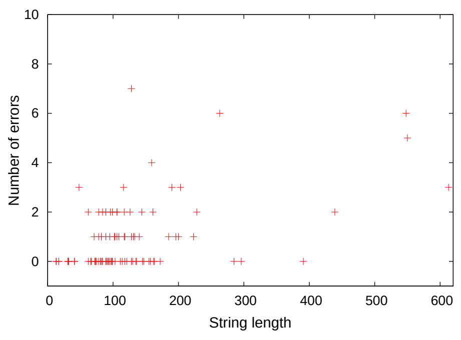
errors, and more than 45% of the cryptograms are solved with three errors or less. The errors that occurred in some of the solved cryptograms were minor ones, all involving spaces only. There are just three examples that showed either 6 or 7 errors, the main reason being that each of these examples had unusual words on particular topics not occurring in the training data.

Variant B produces less errors than other variants. The results show that 59% of the decrypted texts have correctly added spaces with no errors. Furthermore, over 36% of the examples are spaced with just two or one errors, and about 4% with three errors. Just two examples had six errors and all of these are shown in Figure 4.2b.

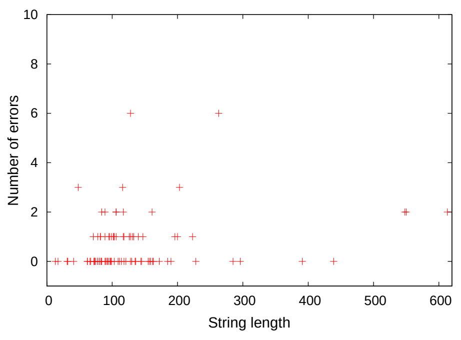
Variant C produces slightly worse results, with just 46% of examples having successfully inserted spaces without any errors with about 45% are spaced with three errors or less. In addition, nine of the solved cryptograms have four errors or more. Figure 4.2c shows the results of variant C. On the other hand, variant D presents very good results, producing similar results to variant B but with a few minor differences.

Figure 4.2e presents the number of errors for variant G as a result of phase two. Clearly the number of errors for each solved cryptogram is much higher, in this case with most of the space insertion errors being greater than 15. Moreover, none of the examples produced no errors and there is just five decrypted texts that were spaced with less than 10 errors.

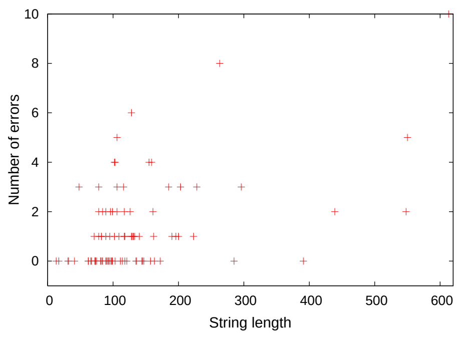
The average number of space addition errors for each variant using both main models, PPMD and PPMC, is presented in the next two tables (Table 4.4 and Table 4.5). These represents the average number of errors for the 90 testing text that have been experimented in this work. Slightly better results are gained from using PPMD models than using PPMC. It is clear that variant B produces the best results although other variants produce good results as well. Again, what is interesting is that the PPM method without update exclusions, which usually does slightly worse at the task of compression, does better here at decryption. The last column in the table presents the number of average errors for variant G. The results show that



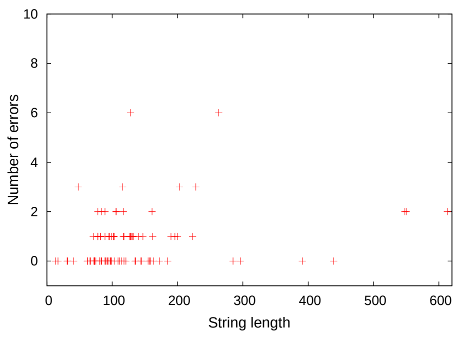
(a) Errors produced from variant A.



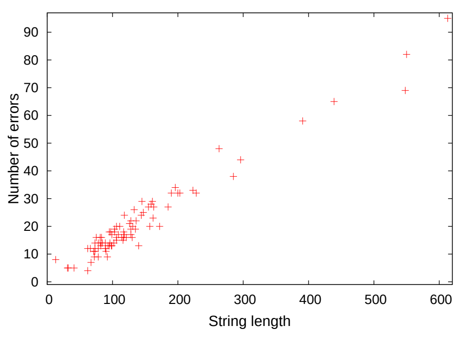
(b) Errors produced from variant B.



(c) Errors produced from variant C.



(d) Errors produced from variant D.



(e) Errors produced from variant G.

Figure 4.2: Segmentation errors produced from the different variants.

Table 4.4: Average number of errors for the phase two variants. (The PPMD model is used for the first four variants.)

Variants	**A**	**B**	**C**	**D**	**G**
Average errors	1.02	0.69	1.30	0.81	21.62

Table 4.5: Average number of errors for the phase two variants. (The PPMC model is used for the first four variants.)

Variants	**A**	**B**	**C**	**D**	**G**
Average errors	1.08	0.71	1.33	0.83	21.62

the Gzip algorithm is not a good way for finding the right solutions with a high average number of errors.

In order to investigate further the accuracy of our spaces insertion algorithms in segmenting the 90 decrypted texts, we used three further metrics: recall rate, precision rate and error rate. Recall is calculated by dividing the number of correctly segmented words (using a compression model) by the total number of words in our original 90 testing texts. The error rate metric is calculated by dividing the number of incorrectly segmented words by the total number of words in the testing texts. Precision is calculated by dividing the number of correctly segmented words by the total number of words which are correctly and incorrectly segmented.

According to Table 4.6, it is clear that the first four variants, which are based on PPMD compression models, achieved very high recall and precision rates for segmenting the 90 decrypted texts. All the errors generated, which are quite low, are those on unusual words not found in the training texts.

The average elapsed time (for variant B) that is required to find the valid decryptions of the transposition ciphertexts with different lengths for different key size is presented in Table 4.7. This table shows the average execution time for decrypting three ciphertexts of different lengths, for both the main

Table 4.6: Recall, precision and error rates for the different variants on segmenting words.

Variants	Recall rate%	Precision rate%	Error rate%
Variant A	95.08	95.08	4.92
Variant B	96.30	96.38	3.70
Variant C	93.91	94.34	6.09
Variant D	95.71	95.91	4.29
Variant G	3.96	16.11	96.04

decryption and spaces insertion phases combined (labelled as 'Both' in the table) and just for the main decryption phase (Phase I). The time which is needed to insert spaces automatically into the decrypted text (the second phase) is based on the execution of Phase II-A (slightly additional time is needed when using Phase II-B). The average execution time in seconds for each ciphertext is calculated by running the program ten times and then calculating the average.

Table 4.7: Average time required to automatically cryptanalysis ciphertexts with different lengths for different keys size.

Ciphertext length (Letter)	**Key size**											
	Time (in seconds)											
	5		**6**		**7**		**8**		**9**		**10**	
	Both	**Phase I**	**Both**	**Phase I**	**Both**	**Phase I**	**Both**	**Phase I**	**Both**	**Phase I**	**Both**	**Phase I**
40	0.72	0.68	0.73	0.69	0.77	0.75	0.97	0.93	2.40	2.38	12.07	12.06
150	1.12	0.70	1.14	0.73	1.20	0.80	1.77	1.35	13.75	10.06	48.07	47.07
300	3.39	0.71	3.62	0.75	3.71	0.87	4.86	1.97	23.01	20.41	95.32	92.41

In summary, the results showed that we are able to achieve 100% success rate as a result of the first phase (Phase I) either by using standard PPM or PPM with-no update exclusions models. We manage to recognise all the plaintexts and solve all the cryptograms in this phase without any errors.

In the second phase, we add spaces to these texts to improve readability by using two methods. The first method is based on a priority queue while the second method uses the Viterbi algorithm to segment words. Our results show that almost all decrypted texts are segmented correctly. The maximum average number of errors (due to space insertions in incorrect places), using PPM compression models, is just slightly > 1.0 (for Variant C). This variant depends on the standard PPM compression model as a basis for calculating the codelength values in Phase II-A. The results showed that by using the Viterbi algorithm (Phase II-B), we can gain slightly better results than the other method (Phase II-A), but it needs slightly more execution time. Variant B showed the best results. This variant depends on using a PPM without update exclusions model using the Viterbi method as the basis for segmenting words.

4.6 Conclusions

We have introduced another use of the compression-based approach for cryptanalysis. A novel universal automatic cryptanalysis method for transposition ciphers and plaintext recognition method have been described in this chapter. Experimental results have shown a 100% success rate at automatically recognising the true decryptions for a range of different length ciphertexts and using different key sizes. This effective algorithm completely eliminates any need for human intervention. In this chapter, we provided pseudo-code for two main phases: automatically decrypting ciphertexts and then automatically achieving readability using compression models to automatically insert spaces into the decrypted texts, as we performed our experiments on ciphertexts for alphabetic English characters, while most previous works did not address this essential problem.

Two compression schemes have been investigated in our research which are Predication by Partial Matching (PPM) and Gzip. The experimental results showed that PPM notably outperforms the other compression scheme Gzip. We also found that both PPM models (PPMD and PPMC) and both PPM variants were able to recognise all the valid decryptions. Concerning automatically adding spaces (word segmentation) afterwards, PPM without update exclusions performs slightly better than the standard PPM method (with update exclusions). Also, slightly better results are obtained from the using of PPMD models rather than PPMC.

The algorithm was able to achieve a 100% success rate using the PPM compression model on different amounts of ciphertext ranging from very short messages (12) to 625 characters, and with different key lengths ranging from 2 to 12. Larger key sizes and longer ciphertext length can be used, but of course it will need longer execution times to perform the decryptions.

Chapter 5

Designing and Evaluating a New Automatic Cryptanalysis of Playfair Ciphers Using PPM

5.1 Introduction

The purpose of this chapter is to explore the use of a compression model for the cryptanalysis of Playfair ciphers. Our new approach to the automatic decryption of Playfair ciphers uses PPM compression to tackle the plaintext recognition problem. This helps to address research question 1 in Section 1.2, and fulfils objectives 2 and 3 that were listed in Section 1.3. We rank the quality of the different plaintexts using the size of the compressed output in bits as the metric. A cryptanalysis of an extended Playfair cipher for a 6×6 grid is also examined. We also use another PPM-based algorithm to automatically insert spaces into the decrypted texts in order to achieve readability; this accomplishes both research question 4 and objective 4 in Chapter 1.

The work in this chapter is based on a conference paper that has been presented at the International Conference on Historical Cryptology (HistoCrypt 2018) and published as part of the Northern European Association for Language Technology (NEALT) Proceedings Series by Linköping University Electronic Press (Al-Kazaz et al., 2018a).

This chapter is organised as follows. The next section covers a discussion of Playfair's weaknesses. Section 5.3 presents related work and prior cryptanalytic methods against Playfair ciphers. Our PPM based method and the simulated annealing search we use for the cryptanalysis of Playfair ciphers are explained in section 5.4. A detailed description of the main principles of the new methodology is provided in this section. Section 5.5 covers the experimentation and results obtained with the conclusions to our findings presented in the final section. The PPM-based cryptanalysis method achieves considerable improvement over previous methods with state of the art performance.

5.2 Playfair's Weaknesses

The Playfair cipher suffers from some major weaknesses. An interesting weakness is that repeated bigrams in the plaintext will create repeated bigrams in the ciphertext. Furthermore, a ciphertext bigram and its reverse will decipher to the same pattern in the plaintext. For example, if the ciphertext bigram "CD" deciphers to "IS", then the ciphertext "DC" will decrypt to "SI". This can help in recognising words easily, especially most likely words. Another weakness is that English bigrams that are most frequently occurring can be recognised from bigram frequency counts. This can help again in guessing probable plain words (Smith, 1955; Cowan, 2008).

Breaking short Playfair ciphertexts (less than 100 letters) without good depth of knowledge of previous messages or with no probable words has proven to be a challenge. Past research has often used much longer ciphertexts—for example, Mauborgne (1914) developed his methods by deciphering a Playfair ciphertext of 800 letters. Also, the Playfair messages that were circulating between the Germans and the British during war had enough depth with many probable words to make them easily readable between these two sides, with no predictor of decrypting success for short messages on anonymous topics (Cowan, 2008). However, the two conditions that the message is short with little depth (no probable words) apply to cryptograms published by the American Cryptogram Association.

5.3 Related Work

Different cryptanalysis methods have been invented to break Playfair ciphers using computer methods. A genetic algorithm was proposed by Negara (2012), where character unigram and bigram statistics were both used as a basis for calculating the fitness function. The efficiency of the algorithm is affected by different parameters such as the genetic operators, ciphertext length and fitness function. Two ciphertexts were examined and decrypted in this paper: one with 520 characters and the other with 870 characters. Hammood (2013) presented an automatic attack against the Playfair cipher using a genetic algorithm. The fitness function calculation was based on character bigram, trigram and four-gram statistics. A ciphertext of 1802 letters was examined in this paper and 22 letters out of 25 were successfully recovered using this method.

Simulated annealing was successful at solving lengthy ciphers as reported by Stumpel (2007). However, he found that short Playfair ciphers of 100 letters or so were unable to be solved. Simulated annealing was also used with a tetragraph scoring function for the automatic cryptanalysis of short Playfair ciphers by Cowan (2008). Cowan managed to solve seven short ciphertexts (80-130 letters) that were published by the American Cryptogram Association.

In summary, several different cryptanalysis methods have been proposed aiming to break Playfair ciphers with varying degrees of success. However, most of these methods were focused on long ciphertexts of 500 letters or more, except Cowan's method (Cowan, 2008). A large amount of information that is provided by long ciphertexts makes breaking them easier while short Playfair ciphers are extremely difficult to break without some known words. In our work, even Playfair ciphertexts as short as 60 letters (without a probable crib) have been successfully decrypted using our new universal compression-based approach. We use simulated annealing in combination with compression for the automatic decryption. Moreover, we have also effectively managed to break extended Playfair ciphers that use a 6×6 key matrix.

5.4 The New Method

This section describes our new method for the automated cryptanalysis of the Playfair cipher. In this chapter, we show how to use the PPM compression-based approach to quickly and automatically recognise the valid decrypt in a ciphertext only attack specifically against Playfair ciphers.

The scoring function should be precisely adapted to the cipher type and its related problem as Lasry (2018) confirmed in his thesis. Moreover, the use of several scoring methods might be required at different stages of the cryptanalysis method to end up with an effective and successful attack. Recovering the first correct key is the most challenging step of an attack especially for hill climbing starting with random keys. These keys often contain a very high number of errors and a large keyspace. At this stage, developing a scoring function that provides better resilience to errors is preferred. After recovering some of the correct keys, a more selective scoring method is more applicable. Higher order n-grams is an example of this method. Thus, it is often necessary to develop an adaptive specialized scoring method that provides a good trade off between resilience and selectivity when the standard scoring methods are not effective Lasry (2018). However, in our adaptive statistical compression approach, the PPM scoring method can handle and tackle all these issues in one effective method. The PPM compression model is used as a single scoring metric for ranking the quality of each putative decryption.

In our experiments described below, we use nineteen novels and the Brown corpus converted to 25 letter English by case-folding to upper case with I and J coinciding for the 5×5 grid and 36 alphanumeric characters for the 6×6 grid to train our models.

Our new method is divided into two main phases. The first phase (Phase I) is based on trying to automatically crack a Playfair ciphertext using a combination of two approaches, which is the compression method for the plaintext recognition and simulated annealing for the search. The second phase (Phase II) is based on achieving readability by automatically adding spaces to the decrypted message produced from phase I, as the spaces are omitted from the ciphertext traditionally.

A variation of an order 5 PPMD model without update exclusions has been used in our experiments for both Phase I and Phase II. This variation is where symbol counts are updated for all contexts unlike standard PPM where only the highest order contexts are updated until the symbol has been seen in the context. In our experiments, this variation has proven to be the most effective method that can be applied to the problem of automatically recognising the valid decryption for Playfair ciphers, but also in other experiments with simple substitution and transposition ciphers.

Simulated annealing is a probabilistic method for approximating the global optimisation of a given function in a large search space. It is a descendant of the hill-climbing technique. This latter technique is based on starting with a random key, followed by a random change over this key such as swapping two letters, to generate a new key. If this key produces a better solution than the current key, it replaces the current one. Different n-graph statistics were used as the scoring function to judge the quality of solutions. After millions of distinct random changes, this technique attempts to discover the correct key.

The weakness of this approach lies in the possibility of being stuck in local optima, where the search has to be abandoned and it is necessary to restart all over again. Simulated annealing (inspired by a process similar to metal annealing) is similar to hill-climbing with a small modification that often leads to an improvement in performance. In addition to accepting better solutions, simulated annealing also accepts worse solutions in order to avoid the local optima. This approach permits it to jump from local optima to different locations in order to find new optima. The probability of the acceptance of the specific solution is dependent on how much the score value is worse. The formula for calculating the acceptance probability is (Cowan, 2008):

$$P_A = \frac{1}{e^{(d/T)}}$$

where e is the exponential constant 2.718, d denotes the difference between the score of the new solution and the score of the current solution, and T is a value called temperature (further details concerning this parameter are described below). Whenever the difference is small, the probability of accepting the new solution is high, while if this solution is much worse than the current one (the difference is large in magnitude), the probability becomes small. The probability value is also influenced by the temperature T. Initially, the algorithm starts with a high temperature value, then it is reduced ('cooled') at each step according to some annealing schedule, until it reaches zero or some low limit. As the temperature drops, the probability of acceptance also decreases and when T is set to zero, the simulated annealing becomes identical to the hill climbing technique.

The main idea of using simulated annealing for the breaking of Playfair ciphers is to modify the current key in the hope of producing a better key. This is based on an approach proposed by Cowan (2008). This can be done by randomly swapping two characters. However, this random change is not enough to effectively break the Playfair cipher by itself. It will usually result in a long search process that often gets stuck within reach of the final solution. So other modifications are needed such as randomly swapping two rows, swapping two columns, reversing the key, and reflecting the key vertically and horizontally (flipping the key top to bottom and left to right). Using a mix of these modifications can lead to the valid solution. For example, swapping two rows will help rearrange rows if they are out of order, as it is very important that rows be in the correct order according to the encipherment rules (Lyons, 2012).

During the whole search process, the hope is that the best plaintext solution that appears is also the correct plaintext. Alternatively, the whole process must be restarted all over again and the value of the temperature should be reset to its original high value (Cowan, 2008). An important aspect of this whole process is the metric that is used to rank the different plaintexts (such as our PPM method). A good metric needs to be able to distinguish effectively between good and poor plaintexts.

Algorithms 6 and 7 present the pseudo code for the first phase of our method. In a preprocessing step prior to the applications of these algorithms, all non-letters including spaces, numbers and punctuation were removed from the ciphertext if a grid of 5×5 is chosen. If a 6×6 grid-width is selected, all non alphabetic letters and numbers were removed from the ciphertext instead. According to selected grid-width, a random key is generated (line 1) and the deciphering operation is initiated using this key. In order to rank the quality of the solutions, the PPM compression method is used by calculating the codelength value for each possible solution (lines 3 and 4). For each iteration, a sequence of changes is performed over the generated key in order to find a solution with a smaller codelength value which represents the valid decryption (lines 5 to 33). The greater the number of iterations, the more likely a solution will be found, but longer execution time will be needed. It is important to note here that we have used negative scores based on the PPM codelengths values in order to maximize rather than minimize scores for the simulated annealing process as per the standard approach adopted in various solutions (Cowan, 2008; Lyons, 2012).

The temperature for the simulated annealing based algorithm is initially set to 20 and reduced by 0.2 in subsequent iterations. (The smaller this amount is, the more likely a solution will be found but this will also result in longer execution time). The initial temperature value is essentially dependent on the cryptogram's length. The shorter the ciphertext, the lower the temperature will be needed and vice versa. We have found in experiments with different length ciperhetexts that for cryptograms of a length of around 70, an initial temperature will need to start at around 10, but for the cryptogram of 700 characters, a temperature at 20 or so is effective.

For each temperature, 10,000 keys are tested then a reduction in the temperature is performed (see lines 9 to 32 in the algorithm). A loop is executed 10,000 times (lines 10 to 31) that modifies the key in the hope of finding a better key with a smaller codelength value. A sequence of different modifications over the key is performed in lines 11 to 17. The encrypted text is then deciphered using the modified key and the codelength value is calculated using the PPM compression method (lines 19 and 20). Then, the difference is calculated between the new codelength value and the previous one. If the new value (line 21) is better (that is, the codelength value is smaller), then the maximum score is set to the new score (line 22), otherwise a probability of acceptance is calculated (line 24) if the temperature is greater than 0 (line 23). In this case, a random number between 0 and 1 is generated, and if the calculated probability is greater than this number, the modified key is accepted (see lines 26 to 27). If we have a new best score, then the old one is replaced (line 29) and systematic rearrangements are performed by calling Algorithm 7.

Algorithm 6: Pseudo code of the main decryption phase (Phase I) for Playfair ciphers.

```
Input  : ciphertext, Playfair grid-width to be either $5 \times 5$ or $6 \times 6$
Output: deciphered-text
1  generate a random key according the Playfair grid-width selected
2  $currentBestKey \leftarrow randomKey$
3  decipher the ciphertext using the currentBestKey and calculate the codelength value
   using the PPM compression method
4  $currentBestScore \leftarrow -$ PPM-codelength score (decipher-text)
5  for $Iteration \leftarrow 0$ to 99 by 1 do
6      $maxKey \leftarrow currentBestKey$
7      decipher and calculate the codelength value using the PPM compression method
8      $maxScore \leftarrow -$ PPM-codelength score (decipher-text)
9      for $Temp \leftarrow 20$ downto 0 by 0.2 do
10         for $Count \leftarrow 0$ to 9999 by 1 do
11             modify maxKey by choose a random number between $(1,50)$:
12                 if the number is 0 then swap two rows, chosen at random
13                 if the number is 1 then swap two columns, chosen at random
14                 if the number is 2 then reverse the key
15                 if the number is 3 then reflect the key vertically, flip top to bottom
16                 if the number is 4 then reflect the key horizontally, flip left to right
17                 if any other number then swap two characters at random
18             $newKey \leftarrow modified\text{-}maxKey$
19             decipher and calculate the codelength value using the PPM compression
               method
20             $newScore \leftarrow -$ PPM-codelength score (decipher-text)
21             calculate $diff \leftarrow newScore - maxScore$
22             if $diff >= 0$ then $\{maxScore \leftarrow newScore; maxKey \leftarrow newKey\}$
23             else if $Temp > 0$ then
24                 calculate $probability \leftarrow \exp(diff/Temp)$
25                 generate a random number between $(0,1)$
26                 if $probability > randomNumber$ then
27                     $\{maxScore \leftarrow newScore; maxKey \leftarrow newKey\}$
28             if $maxScore > currentBestScore$ then
29                 $currentBestScore \leftarrow maxScore$; $currentBestKey \leftarrow maxKey$
30                 Make systematic rearrangements(ciphertext, currentBestKey,
                   currentBestScore)
31         end
32     end
33 end
34 return the deciphered text with the best key
```

Algorithm 7: Make systematic rearrangements

```
   Input  : ciphertext, currentBestKey, currentBestScore
   Output: currentBestKey, decipher-text
 1 $flag \leftarrow true$
 2 while flag do
 3     $flag \leftarrow false$
 4     perform systematic mutations over the currentBestKey:
 5         decipher and calculate the codelength value using the PPM compression method
 6         $newscore \leftarrow -$ PPM-codelength score (decipher-text)
 7         if $newscore > currentBestScore$ then
 8             $flag \leftarrow true$
 9             $currentBestScore \leftarrow newScore$; $currentBestKey \leftarrow newKey$
10             continue outer While loop
11     perform systematic row-swaps and column-swaps over the currentBestKey:
12         decipher and calculate the codelength value using the PPM compression method
13         $newscore \leftarrow -$ PPM-codelength score (decipher-text)
14         if $newscore > currentBestScore$ then
15             $flag \leftarrow true$
16             $currentBestScore \leftarrow newScore$; $currentBestKey \leftarrow newKey$
17             continue outer While loop
18     perform swapping of four characters:
19         decipher and calculate the codelength value using the PPM compression method
20         $newscore \leftarrow -$ PPM-codelength score (decipher-text)
21         if $newScore > currentBestScore$ then
22             $flag \leftarrow true$
23             $currentBestScore \leftarrow newScore$; $currentBestKey \leftarrow newKey$
24             continue outer While loop
25 end
26 return currentBestKey, decipher-text
```

These include mutations (lines 4 to 10 in the new algorithm, Algorithm 7), row swapping and column swapping (lines 11 to 17) and an exhaustive search over all 4! possible permutations of each group of four symbols (lines 18 to 24). Swapping single pairs of letters results in the search getting stuck in local maxima too often, so we added the swapping of all possible combinations of 4 symbols to try to avoid that. Trying 3, 5, or even more combinations of symbols is possible, but of course the higher the number, the search starts getting very expensive, so 4 provides a reasonable compromise. Finally, the deciphered text is returned with the smaller codelength value which represents the best solution found (line 34, Algorithm 6). This has proved adequate for the solution of most ciphers, but if necessary, it is still possible to iterate the attack several more times.

Concerning the second phase of our approach, Algorithm 8 illustrates the pseudo code for this phase ‘word segmentation phase’. PPM is again applied to rank the solutions. Two alternative methods have been investigated in this phase (the same as for the transposition ciphers experiment in the previous chapter). However, only the Viterbi method is presented here as it showed a slightly better result than the other method. The Viterbi algorithm is used in this phase to find the best possible segmentation. In this algorithm, looping over the deciphered text (that was produced as output from Algorithm 6) is performed in line 2. A word segmentation algorithm based on the Viterbi algorithm is then used to search for the best performing segmentations to keep in a priority queue, and those which showed poor codelength values are pruned (see lines 3 to 5). The best segmented deciphered text is returned in the last line (line 6).

Algorithm 8: Pseudo code of Phase II for Playfair ciphers

```
  Input  : the deciphered text from Phase I
  Output: segmented deciphered text
1 maximum size of Q1 (priority queue) ← 1;
2 do
3     use the Viterbi algorithm to search for the best segmentation sequences;
4     store the text that have the best segmentation which present in Q1;
5 while the end of the deciphered text;
6 return the best segmented deciphered text from Q1;
```

5.5 Experimental Results

In this section, we discuss the experimental results of our approach. As stated, in our method the order 5 PPMD model has been trained on a corpus of nineteen novels and the Brown corpus using 25 English letters (when a 5×5 grid is used) and 36 alphanumeric characters (when a 6×6

grid is used). Regarding the cryptograms test corpus, 75 different cryptograms were chosen at random from different resources including cryptograms published by the American Cryptogram Association, cryptograms published by geocache enthusiasts, Royal NZ navy historical cryptogram and examples from Mauborgne (1914), Gaines (1956), Friedman and Lambros (1985), Singh (2000) and Negara (2012). Samples of which are listed in Appendix III. Cryptogram lengths ranged from 60 to 815 letters.

A sample trace of a decryption is shown in Figure 5.1 for the cryptogram: '`dohrxnwpscqusfrwchrnpctsehagvpstsfaprdtuipwolacgqupfwptslaqsizbedxq-usfwscosfraevstngqu`'. This shows the best score as it changes during the execution of Algorithm 7 for the main decryption phase. The scores are increasing (i.e the codelengths are decreasing). The solution of this ciphertext is a proverbial wisdom that has been attributed to Damon Runyon: "*It may be that the race is not always to the swift nor the battle to the strong but that is the way to bet*". This ciphertext is one of the short cryptograms (82 character long) that have been published by the American Cryptogram Association, which usually publishes 100 ciphertexts every two months including one or more Playfair ciphers, as a challenge to its members (Cowan, 2008). Cowan has stated that it is extremely difficult to break short messages of 100 letters or so, especially when there are no suspected probable words or cribs and very little depth of knowledge of previous messages. However, our method is able to solve the following examples in addition to the other cryptograms that were listed by Cowan as well as even shorter ciphertexts of 60 letters or so.

```
Iteration:          35
Mutation gain:      -235.55  rkdavmetontxtherndacobotalwaystothescruflqrthemrtxt-
                    letothestfmpbiktxthatokthewaytomatx
                    key: znmbcwagerfhilduvxyktsqpo
Mutation gain:      -232.89  rkdavmetontxtherndacobotalwaystothescrufbtrthemrtxt-
                    letothestmfpbzktxthatokthewaytomatx
                    key: inmbcwagerfhzlduvxyktsqpo
Mutation gain:      -230.84  ridavmetontxtherndacobotalwaystothescrufyorthemrtxt-
                    letothestxdpbzitxthatoithewaytomatx
                    key: knmbcwagerfhzlduvxyitsqpo
Mutation gain:      -226.48  ridavmetontxtherndacobotalwaystothescrufyorthemrtxt-
                    letothestucpbkitxthatoithewaytomatx
                    key: znmbcwagerfhklduvxyitsqpo
2-Mutation gain:    -225.29  ridayketobtxtherbdeconotalwaystothescrufyorthekrtxt-
                    letothestucnamitxthatoithewaytoketx
                    key: zbkncwagerfhmlduvxyitsqpo
Mutation gain:      -221.66  ridaybetoktxtherkdeconotalwaystothescrufyorthebrtxt-
                    letothestucngmitxthatoithewaytobetx
                    Key:zkbncwagerfhmlduvxyitsqpo
Mutation gain:      -220.15  ridaybetvstxthersaeksnotalwaystotheskrufyorthebatxt-
                    letothestukngmitxthatksthewaytobetx
                    Key: zcbnkwagerfhmlduvxyitsqpo
Mutation gain:      -215.91  ridaybetvstxthersaemsnotalwaystothesmrufyorthebatxt-
                    letothestumngkitxthatmsthewaytobetx
                    Key: zcbnmwagerfhklduvxyitsqpo
Mutation gain:      -207.60  rmdaybetvstxthersaeisnotalwaystothesirufnorthebatxt-
                    letothestningkmtxthatisthewaytobetx
                    Key: zcbniwagerfhklduvxymtsqpo
Mutation gain:      -204.66  itzaybetvstxthersaeisnotalwaystotheswiufnorthebatxt-
                    letothestorngbutxthatisthewaytobetx
                    Key: dcbniwagerfhklzuvxymtsqpo
Mutation gain:      -195.04  itmaybetvstxthersaeisnotalwaystotheswiufnorthebatxt-
                    letothestomngbutxthatisthewaytobetx
                    Key: dcbniwagerfhklmuvxyztsqpo
Row-swap gain:      -184.98  itmaybethvtxthervaeisnotalwaystotheswiftnorthebatxt-
                    letothestzongbutxthatisthewaytobetx
                    Key: dcbniwagerfhklmtsqpouvxyz
Row-swap gain:      -162.46  itmaybethatxtheraceisnotalwaystotheswiftnorthebatxt-
                    letothestrongbutxthatisthewaytobetx
                    key: dcbnifhklmtsqpouvxyzwager
```

Figure 5.1: Example trace of decryption of a cryptogram of 82 letters from the American Cryptogram Association.

A second example in Figure 5.2 illustrates the robustness of our compression approach by showing how it is able to solve a very short cryptogram. The ciphertext is a 60 letter sentence (a quote by Garrison Keillor): *Cats are intended to teach us that not everything in nature has a purpose.* The best solution for this example is '`catsareintendedtoteachusthatnotexerythinginxnaturehaoapurposew`' with the best codelength value `-137.68` resulting in only two errors: `x`$\rightarrow$`v` in '`exerything`' and `o`$\rightarrow$`d` in '`hao`'.

```
Iteration:          89
                    -164.32   catsareintencectoteakiusthhonotexerythinginxnature-
                    hatapurposid
Mutation gain:      -161.21   catsareintencectoteakiusthaonotexerythinginxnature-
                    hatapurposid
Mutation gain:      -160.36   pltsapeintencectoteadbusthahnotexerythinginxnature-
                    oatapurposid
Mutation gain:      -159.28   pltsapeintendedtoteacbusthahnotexexbthinginxnature-
                    oatapurposic
Mutation gain:      -156.08   datsareintendedtoteakiusthaonotexexfthinginxnature-
                    hatapurposic
Mutation gain:      -155.29   katsareintendedtoteakiusthatnotexexythinginxnature-
                    haoapurposic
Mutation gain:      -154.12   ratsakeintendedtoteakiusthatnotexeocthinginxnature-
                    hasapukposic
Mutation gain:      -150.31   ratsileintendedtotealausthatnotexeokthinginxnature-
                    hasapudiospc
2-Mutation gain:    -149.97   ratsileintendedtotealhusthatnotexevcthinginxnature-
                    hasapudiosev
Mutation gain:      -143.16   ratsaceintendedtoteachusthatnotexelvthinginxnature-
                    hasapucposev
Mutation gain:      -138.52   catsareintendedtoteachusthatnotexerythinginxnature-
                    haoapurposev
Mutation gain:      -137.68   catsareintendedtoteachusthatnotexerythinginxnature-
                    haoapurposew
```

Figure 5.2: Example trace of decryption of a short cryptogram of 60 letters.

A third example is a Royal New Zealand navy historical case of length 102 letters: `KXJEY UREBE ZWEHE WRYTU HEYFS KREHE GOYFI WTTTU OLKSY CAJPO BOTEI ZONTX BYBWT GONEY CUZWR GDSON SXBOU YWRHE BAAHY USEDQ.`

On the second of August, 1943, Japanese destroyer Amagiri rammed and sank the American patrol boat PT-109, which was under the command of US Naval Reserve Lieutenant and future President John Kennedy. After arriving ashore, Kennedy sent this ciphertext, which was encrypted using a Playfair cipher (Klima and Sigmon, 2012). The execution trace is shown in Figure 5.3.

```
Iteration:          12
                    -310.41 dteoatoniogeninevostinactionineaacketxtstraitxtkombri-
                    sswderesucopeuwrewoftgeanixrequistanyinformatbohx
Mutation gain:      -305.07 dteoatonioveninegostinactioninaracketxtstraitxtkombri-
                    sswderesucopexyrewoftvegaixrequistanyinformatbohx
Mutation gain:      -303.05 dteoatonioveninepostinactioninbracketxtstraitxtkomari-
                    sswderesucogeqcrewoftveqhixrequistanyinformatbohx
Mutation gain:      -299.99 dteoatoniogeninepostinactionineaacketxtstraitxtkomari-
                    sswderesucoveuwrewoftgeanixrequistanyinformatbohx
Mutation gain:      -291.27 dteoatonioleninepostinactioninbpacketxtstraitxtkomari-
                    sswderesucovexcrewoftlepvixrequistanyinformatbohx
Mutation gain:      -290.34 dteoatoniopeninelostinactioninblacketxtstraitxtkomeai-
                    sswderesucovexcrewoftpelvixrequistanyinformatbohx
Row swap gain:      -267.17 dtboatoneopeninelostinactioninblacketxtstraitxtwomeae-
                    sswderesucovexcrewoftpelvixrequestanyinformatiohx
Mutation gain:      -262.98 ptboatoneoweninelostinactioninblacketxtstraitxtwomile-
                    sswperesucovexcrewoftwelvexrequestanyinformationu
```

Figure 5.3: Example trace of decryption of the 102 letter Royal NZ navy historical ciphertext.

The following example is a puzzle cryptogram of 96 letters from the geocache world (https://bcaching.wordpress.com/2008/08/08/puzzles-part-3/): ‘`sa cb av hm ka do st th ps mn qs fr hm sx bt su tw tg wg mh mc ok sd oz ts fy tw ts vc ec gs gt wl dl sr oz tb tl ps tg ex cm co dl kh wl wg mh ex av`’. Figure 5.4 presents the intermediate results and the final solutions produced by each iteration for this cryptogram. According to this example, iteration 89 produced the best solution with the best score with a compression codelength value of 215.81 and is the valid decrypt.

```
Iteration:  0
Mutation gain:    -424.28  oftmxozhxkftpoobinzifiqazhyapenfvdvbgyhzirlapflhopw-
                  svdoptorthybvuvutialhepcvinvbreriolutazuvgyhzrexo
...
Mutation gain:    -311.85  tuemuirecolstaurytrtforxreafmstoopanilercekqbsulaty-
                  dopatiekedanalondtfulsmonytancheckandeqloilerchui
Iteration:  1
Iteration:  2
                  -311.01  adpcdhowwhsvalarcaucedofowleyldmailiumwoseheatrelar-
                  bailaonmemlilgatstorelylscalikeesectswpgaumwokedh
Iteration:  3
Iteration:  4
Iteration:  5
...
Iteration:  24
                  -309.15  hkxepmmaskbitesratokhdtvmabasedaeferelamlamntbinetn-
                  refetlpgwhereeceithinesevaterbcalxleismecelambcpm
...
Iteration:  30
Row swap gain:    -304.26   amsegplaysonasarealmcarblaterstcrustitalsniymeinsa-
                  hirusaezkstatsorodtbinsrmreastpensnkodyboritalpegp
...
Iteration:  89
Row swap gain:    -215.81   thecoxordinatesarenorthfortydegrexeszeropointfiveth-
                  rexetwowestseventyfivedegreestwopointfivezerotwox
...
Iteration:  100
```

Figure 5.4: Solutions produced during selected iterations for a puzzle cryptogram of 96 letters.

Our method was also able to solve a 6×6 Playfair cipher with a few minor errors. The next sample is a cryptogram that was posted on a puzzles forum originating from geocache enthusiasts (http://members2.boardhost.com/barryispuzzled/msg/1500564217.html):

```
lpqtj zfpvf ndsvb joamd j4mva nrfeu nbhis nhcru chhfs otb1e kbueg
qejtv kgscn kq3ez kgwix eavej nstda usbfj cvkgs cbtqz 5nmqa nc0jc
xeiue nhtnb rbwhg krabz 1j0mn dierf rabfq vjvkf dnrbk nkd0u cbwhn
dsdlv vpvha gvucb bnyjc vtzpf brrab gtmqa j4fmt ryjbu vldtq fsnts
awfhl pvthc 0hpvv obvmd jvra7 zhfew irgh3 8qpck r5z7r gaw1k biyjg
h3w8w qfau3 v7dra bfnbe jgkke kvhfc 0dsjy raghx bjbqm bhhgc hnwyj
bxgkk ekvhf dcvjo uebxr rsch1 jwmvu bxlbi nmraw ckbnh g1jrn dtchl
vfpur raihj 4fmoq jbrbj zfmtr yjbur acjck vughh tchjv rauxc hrzkc
hchff elnob mvvjh cgerf rgawu xchrz uxvfb fcqbt mdfjh chgrf sujep
qbrej yjrgy jchmv eseue ckgau ejrbr uvlej mq1jv mh0mv txhz
```

Part of the execution trace is shown in Fig. 5.5.

```
Iteration: 1
-1604.59   jolxlygoodandwelldoneyouhavecrackedanextendedplayfaircipherusinga-
sixbysixgridtheadvantageofusingall30slphanumericxcharactersisthatyoucangivet-
hecoxordinatesasnumbersandnotwordsbutbecarefultogetthefullkeycorxrecttoavoid-
awastedjourneynowofftonorthztdegrexesx3poinwytzwest3degrees5x9xoint1x2atheca-
cheishidxdenunderthestilepleaseensureitishidxdenfromviewxwhenyouputitbackjus-
tincaseyouhavenotgotxthekeycompletelycorxrecttheminutesarenorthtwentyninedec-
imalfouronefivewesttwentytwodecimaloneonesevenwehopetherewereenoughcribsinth-
etexttohelpyouonyourway

-1594.57   jolxlygoodandwelldoneyouhavecrackedanextendedplayfaircipherusinga-
sp5bysixgridtheadvantageofusingall50klphanumericxcharactersisthatyoucangivet-
hecoxordinatesasnumbersandnotwordsbutbecarefultogetthefullkeycorxrecttoavoid-
awastedjourneynowofftonorthztdegrexesw3poinxytzwest5degreeswp3point1w2atheca-
cheishidxdenunderthestilepleaseensureitishidxdenfromviewxwhenyouputitbackjus-
tincaseyouhavenotgotxthekeycompletelycorxrecttheminutesarenorthtwentyninedec-
imalfouronefivewesttwentytwodecimaloneonesevenwehopetherewereenoughcribsinth-
etexttohelpyouonyourway
```

Figure 5.5: Example solutions produced for a 6×6 Playfair cryptogram.

The experimental results of Phase I, when the order 5 PPM method without update exclusions is used, showed that most of the cryptograms are successfully decrypted with no errors. Table 5.1 presents the results from testing ciphertexts for various lengths. The results overall showed that we are able to attain very high success rates and 65 ciphertexts out of 75 were efficiently solved. Also, 100% of ciphers of length greater than 120 were decrypted. The results achieved have outperformed the previous state-of-the-art results (Cowan's results).

Table 5.1: Results when testing ciphertexts with different lengths.

Ciphertext Length	**60-79**	**80-99**	**100-119**	**120-149**	**150-199**	**200-815**
No. of Ciphertexts	9	21	16	11	8	10
Success Rate (%)	67	81	81	100	100	100

Referring to the second phase of our method, as the spaces are omitted from the ciphertext traditionally, this phase focuses on segmenting the decrypted messages that are outputted from the first phase. The edit distance (or Levenshtein distance) metric is used to evaluate how the decrypted message is differentiated from the original message by counting the minimum number of the removal, insertion, or substitution operations required to transform one message into the other (Levenshtein, 1966). In almost all cases, the correct readable decryptions were efficiently found as illustrated in Figure 5.6.

The number of space insertion errors for each testing cryptogram is plotted in Figure 5.7. We can see that the number of errors for most cryptograms are very low and the correct segmentations are obtained in most cases. The average space insertion errors for the ciphertexts that were experimented with in Phase II is less than one error.

Ciphertext	`byntlbneonnuimmzqnhpbkxnqmfqoqnmugclqmeuersuqpnzigqbqyipilqtku`
Decrypted text	`cats are intended to teach us that not exerything in nature ha o a purpose`
Ciphertext	`kuinbrnuikcnqmhuvgtnnmybkgbromruknqmmnknqdmpvgniignkoneumokgpgxytqsu`
Decrypted text	`experience is the worst teacher it gives the test before presenting the lesson`
Ciphertext	`pqghqncnndqyhfqugqqmeusxqmfqdpkgqbitqdkunurqioinnlpgvqvpbmlwhuqoimigbzka`
Decrypted text	`a n egotist is a man who thinks that if he hadnt been born people would have wondered why`
Ciphertext	`qmghblxytkyfihogkunugiqoqmgnqmgincimtlqmmpnuikiwszqmgiknliqrbhafgtigtldnnqgtxz`
Decrypted text	`the grass may be greener on the other side of the fence but there s probably more of it to mow`
Ciphertext	`hmfnuwfntufdbgushmtuqmckqnutfpmatuzfmbfntylxqpthrkucnrkrmcqdamibarurntumucoffdummbrnki`
Decrypted text	`the likeliness of a thing happening is inversely proportional to its desirability fin agles first law`
Ciphertext	`dohrxnwpscqusfrwchrnpctsehagvpstsfaprdtuipwolacgqupfwptslaqsizbedxqusfwscosfraevstngqu`
Decrypted text	`it may be that the race is not always to the swift nor the battle to the strong but that is the way to bet`

Figure 5.6: Example of solved ciphertexts with spaces inserted after Phase II.

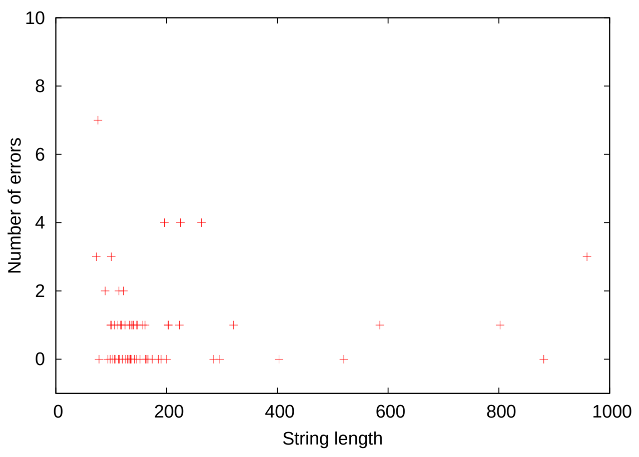


Figure 5.7: Segmentation errors produced as a result of the Phase II algorithm.

Table 5.2 lists the high recall and precision rates and the low error rate produced by our segmentation algorithm.

Table 5.2: Recall, precision and errors rates for our method for word segmenting the decrypted output produced from Phase I.

Recall (%)	**Precision (%)**	**Errors (%)**
97.19	96.7	2.81

The execution times required to decrypt a number of Playfair ciphertexts by our method are presented in Table 5.3. This table shows the decryption time in seconds for Phase I of our method. The results indicate that our method produces reasonable decryption times, and in most cases the successful decrypts of longer ciphertexts were obtained after only one or two iterations.

Table 5.3: Decryption times for Phase I for different ciphertexts.

Ciphertext Length *(Letter)*	**60**	**71**	**86**	**100**	**124**	**185**	**235**	**526**	**730**
Time *(Sec)*	457	539	507	93	36	17	135	107	101

5.6 Conclusions

An automatic cryptanalysis of Playfair ciphers using compression has been introduced in this chapter. A new effective method has been developed for a number of challenging classical cryptographic cases. In particular, a combination of simulated annealing and PPM compression was used in the automatic decryption method. The compression scheme was found to be an effective method for ranking the quality of each possible permutation as the search was performed. In 65 of the 75 ciphertexts that were experimented with (without using a probable word) for different lengths (from as short as 60 letters up to 815), almost all the correct solutions were found. The exception was just a few very short ciphers which resulted in one or two minor errors in the decrypted output. Moreover, we have also managed to decrypt an extended Playfair cipher for a 6×6 key matrix.

In addition, a compression-based method was used to segment the decrypted output by insertion of spaces in order to improve readability. Experimental results show that the segmentation method was very effective producing on average less than one space insertion error with a recall and precision of over 96% for the ciphertexts that were tested.

Chapter 6

Automatic Cryptanalysis of Classical Arabic Ciphers Using Compression

6.1 Introduction

The previous chapters investigated the automatic cryptanalysis of different ciphers for the English language, and then the PPM compression-based method used was found to be very effective. This chapter seeks to address an open question, which is related to one of our main research questions (research question 3) in Chapter 1, which was to find out whether this new approach is applicable to other languages. This question is important because a solution for one language does not necessarily mean it will be a corresponding solution to other languages, especially for non-related languages. In order to overcome this gap in the research related to our cryptanalysis approach, we need to apply this new approach to another non-related language. We choose the Arabic language specifically because it has characteristics that differentiate it from other languages and because Arabic has a rich morphology which presents challenges for natural language processing and cryptographic systems. Furthermore, we choose this language specifically to ascertain whether the Arabic language provides a greater challenge for decryption.

Despite the large number of Arabic speakers and with Arabs being among the first to use encryption systems as well as having success in breaking them, unfortunately encryption systems and cryptanalysis methods for Arabic are very few in comparison to other languages such as English. The purpose of this chapter is to generate three main Arabic ciphers (substitution, transposition, Playfair) and explore the use of compression models for the automatic cryptanalysis of these ciphers; this fulfils objective 5 in Section 1.3. Using compression schemes adapted for the Arabic language as one way to tackle the Arabic plaintext recognition problem, as we do in our work, is a completely new approach and not tried before in the literature. The automatic cryptanalysis of these ciphers is the first step to solve some historical Arabic ciphertexts. In addition, further algorithms also based on using an adapted PPM compression model of Arabic text are also introduced to automatically insert spaces to the decrypted texts in order to achieve readability. This helps to address both research question 4 and objective 4 that were outlined in Chapter 1.

In this chapter, different Arabic corpora used in our experiments are described in section 6.2. In order to measure the effectiveness of compression models and to facilitate the solution of short cryptograms using our method, the entropy of Arabic is investigated in Section 6.3. An overview of related work on the cryptanalysis of Arabic ciphers is presented in Section 6.4. Samples of simple substitution, transposition and Playfair ciphers using Arabic language are reviewed as well in this chapter. Cryptanalysis methods for the three different Arabic ciphers and their experimental results are described. The conclusions are stated in the last section.

6.2 Arabic Language Corpora

A corpus can be defined as a body of structured texts that is typically stored in machine-readable form for the purpose of statistical analysis (Teahan, 1998). Different Arabic corpora are introduced in this section in order to explore the most effective corpus to be used in our cryptanalysis methods for the Arabic language ciphers. These corpora are varied to include classical, modern and mixed Arabic texts. Classical Arabic is the language of the primary religious book of Islam–Holy Quran–(1,400-year-old) and other ancient books from that epoch (Dukes and Habash, 2010), while modern Arabic is the language of most of the current Arabic publications and printed Arabic media (Alghamdi and Teahan, 2017). In terms of corpora size, our generated corpora range from small to large encoded using a UTF-8 scheme.

Concerning the classical Arabic text, three different corpora have been set up with different sizes (small, medium and large) as presented in Table 6.1. These corpora consist of various classical text files of different genres that have been extracted from the BACC corpus (Teahan and Alhawiti, 2013). The Bangor Arabic Compression corpus (BACC) is a 31-million words corpus collected from different sources such as magazines, books and websites. This corpus contains texts from a wide range of genres containing both classical and modern Arabic.

Three different corpora with different sizes and genres are also constructed to represent modern Arabic text. Further details are presented in Table 6.1. These modern corpora are derived from Corpus A (Alkahtani, 2015). Corpus A is a modern corpus recently published with different genres which covers several current modern standard Arabic areas. This corpus is collected from the bilingual newspaper Al-Hayat website, and from the open-source online corpus, OPUS. It consists of 27-million Arabic words.

Referring to the last corpora type (Mixed), classical and modern Arabic texts are used to produce the corpora. The small and the medium size Mixed corpora are elicited from both the BACC and the Corpus A. The large Mixed

Arabic corpus is a combination of the BACC, Corpus A and a selection of files from the King Saud University Corpus of Classical Arabic (KSUCCA) (Alrabiah et al., 2013). KSUCCA is a corpus made up of classical Arabic texts with the purpose of studying the lexical semantics of the Holy Qu'ran. It is a 50-million word corpus covering several genres. The overall file size of our large Mixed Arabic corpus is 513,291,607 bytes and consists of 58,068,493 words. Further details concerning the large Mixed Arabic corpus are shown in Table 6.2.

Table 6.1: Different examined corpora.

Corpus type	**Category**	**Size in bytes**	**Description**
Classical	Small	31,018,170	Collection of classical books and novels
	Medium	201,693,734	
	Large	252,159,806	
Modern	Small	28,362,790	Collection of modern books and novels
	Medium	149,984,803	
	Large	252,338,294	
Mixed	Small	98,663,231	Collection of classical and modern books and novels
	Medium	257,172,190	
	Large	513,291,607	

Table 6.2: Details of sub-corpora used to construct the large Mixed Arabic corpus used to train the PPM models.

Corpus name	**Size in bytes**	**List of topics**
BACC	256,867,247	sports, culture, economics, education, art and music, political literature and heritage, history, religion
Corpus A	252,338,294	business, cinema, opinions, conferences, economics, politics
KSUCCA	4,086,066	religion, linguistics, literature, science, sociology, biography

The most important step in our implementation, as stated, is the training step. During the training phase, a large set of training text is used to prime the models. Training text is chosen that is hopefully representative of the text being compressed and consequently better able to predict it. Experiments with English text show that training can improve performance dramatically as when skipping the training phase, there is not enough and sufficient data at the beginning to effectively compress the texts. PPM's performance substantially improves when large amounts of priming text is used (Teahan and Cleary, 1996). In our experiments, all previously mentioned corpora are used to train the models.

6.3 The Entropy of Arabic using PPM

In order to deal with Arabic texts, in which each character needs two bytes to be represented, an adaptive PPM compression model for Arabic language ("CSA-PPM", as discussed in Section 2.5.5) has been explored. Another encoding method commonly used is also investigated as a comparison, which is the Buckwalter Arabic transliteration. It is an ASCII transliteration system that follows the standard encoding schemes to represent Arabic texts for computers. In this transliteration scheme, Arabic orthographical characters are strictly substituted one to one. For example, the Arabic word "كتاب", which means "book", is represented by the letters "ktab" (Habash et al., 2007). When this transliteration is used as a pre/post-processing step with PPM compression, we call this method "BA-PPM".

The performance of the "CSA-PPMD" and the "BA-PPMD" methods is compared with PPMD on raw Arabic text in Table 6.3. The table shows the Arabic encoding methods and the compression ratios for the trained PPM using order 5 models. Previous experiments have shown that order 5 models perform best at compressing the Arabic text (Alhawiti, 2014; Alkahtani, 2015). The text was converted to 37 character Arabic (36 Arabic letters and space). In this case, the training text used was the classical and modern Arabic texts that were used to produce the large Mixed Arabic corpus. The raw Arabic text file positions and its equivalent size in characters for the CSA-PPM and BA-PPM methods are presented in the first and third columns of the table. The codelength values in bytes are shown in columns two and four. Column five provides the codelength ratios for PPM on raw Arabic language, while column six exhibits the codelength ratios in bits per character for both the CSA-PPM and BA-PPM methods. This table shows that the best result occurs for both the CSA-PPM and BA-PPM methods. The importance of using these two adaptive PPM methods for the Arabic language is clearly illustrated in this table with a significant improvement in the codelength compression ratio, as presented in the last column of the table (and in Figure 6.1). The results also indicate that the compression ratio produced by both the BA-PPM and CSA-PPM methods are exactly the same. (For this reason, we will keep on using the CSA-PPM method in the upcoming experiments).

Figure 6.1 illustrates the reduction in the codelength ratio versus file position for the PPMD on raw Arabic language and CSA-PPMD methods. This is a measure of how well the different compression methods would perform at compressing the same text. The figure shows that there is a significant improvement in the performance of the CSA-PPM method over the performance of the standard PPM for the raw Arabic text. The compression for the CSA-PPM model generally increases to reach the best compression ratio at 1.923 bpc at file position 10000000, followed by a steady increase in the compression ratio. This is because in the training data, several corpora were concatenated together to create the training text and it is at this point where one corpus completes and a new one starts.

Using trained PPM with character-based encoding on the large Mixed Arabic corpus of size 513 megabytes results in an estimate of 1.923 bits per character. This compares with 1.812 bits per character for the English text obtained by concatenating several small corpora (the Brown Corpus, the Wall Street Journal, the LOB Corpus, the King James Version of the Bible and the complete works of Jane Austen) of size 35 megabytes (Teahan, 1998).

Table 6.3: Comparing compression results between PPMD on Raw Arabic language, CSA-PPMD and BA-PPMD methods.

Raw file position (input) (byte)	Raw PPM codelength (output) (byte)	CSA/BA-PPM size (input) (char.)	CSA/BA-PPM codel. (output) (byte)	Raw PPM codelength ratio (bpc)	CSA/ BA-PPM codelength ratio (bpc)	Improv-ements (%)
500	177	273	152	5.187	4.454	14.13
1000	331	545	298	4.859	4.374	9.98
5000	1379	2743	1299	4.022	3.789	5.79
10000	2561	5502	2414	3.724	3.510	5.75
50000	11246	27628	10338	3.256	2.993	8.08
100000	21422	55298	19483	3.099	2.819	9.04
500000	98807	276718	88229	2.857	2.551	10.71
1000000	200092	554274	173946	2.888	2.511	13.05
5000000	888126	2783077	711847	2.553	2.046	19.86
10000000	1699941	5573232	1339915	2.440	1.923	21.19
50000000	9290505	27742681	7389196	2.679	2.131	20.46
100000000	19123851	55405677	14993021	2.761	2.165	21.59
500000000	99643963	274864694	74594365	2.900	2.171	25.14

6.4 Related Work

Despite the fact that the science of cryptology was born among the Arabs and they managed to break the mono-alphabetic substitution cipher after many years of its successful use (Singh, 2000), only some publications have investigated the use of the Arabic language for encryption/decryption systems and none for the problem of solving transposition and Playfair ciphers. For example, Scharwächter and Vogel (2015) used Hidden Markov Models (HMMs) for the problem of solving simple substitution ciphers for Arabic optical character recognition (OCR). Bigram and trigram character language

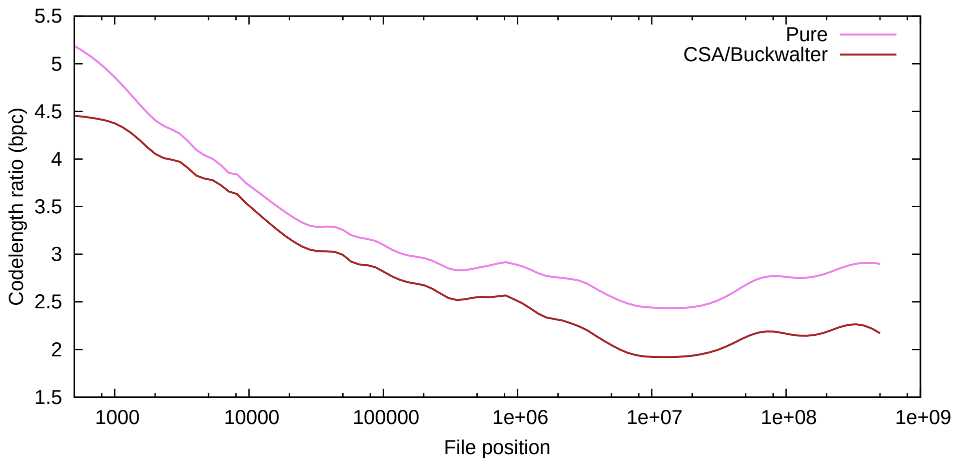


Figure 6.1: The reduction in the codelength ratio versus file position for the PPMD on raw Arabic language and CSA-PPMD/BA-PPMD texts.

models were used. Two different Arabic documents were examined in this work: the first 500 lines of the Quran (42,170 characters) and the first page of a book about the Arabic language (18 lines, 700 characters) provided by the Arabic Language Technology Centerl (ALTEC). Results showed that the proposed algorithm worked well in deciphering these two documents.

Alqahtani et al. (2013) explored the use of the Vigenère cipher on mod 39 as a new approach to Arabic encryption and decryption. They adapted the classical cipher of modular 26 and altered it with modular 39 according to the 28 Arabic letters, space and digits. Habeeb (2016) presented an algorithm that uses genetic algorithms to cryptanalyze an encrypted Arabic text using the Vigenère cipher. Character unigram and bigram statistics were both used as a basis for calculating the fitness function. Different ciphertext lengths (400, 600, 1000 letters) and different key lengths (5, 10, 20) were experimented with. A success rate of 90% was achieved when a key length of 10 letters was used to crack a cryptogram of 1000 letters. In addition to other publications which also investigated the use of the Arabic language for encryption/decryption systems (Rihan and Osma, 2016; Muanalifah, 2017; Shaban and Najimaldin, 2017).

In our approach, Arabic substitution, transposition and Playfair ciphers are generated and then new adapted methods using compression for the automatic cryptanalysis of Arabic ciphers are proposed. As far as we know, this is the first work to demonstrate an effective automatic cryptanalysis for transposition and Playfair ciphers in Arabic. Various Arabic ciphertexts with different lengths, from very short ciphers to ciphers of length 750 letters, are then tested in our experiments described below.

6.5 An Automatic Cryptanalysis of Arabic Simple Substitution Ciphers

As stated in Section 2.2.5.1, in a simple substitution each character of the alphabet is replaced by another predetermined letter of the alphabet. For example, using the substitution:

Plaintext letter : ا أ إ آ ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه ة و ؤ ي ى ئ ء

Ciphertext letter: ط س ة ي ق ء ث ج غ خ د ص ؤ أ ك ش ر ئ ض ا ظ ع ح ف ب ز ل م ن ه إ و ذ آ ى ت

the message ‘أنظمة التشفير’ would be encoded as ‘سماله طزءشحذؤ’.

In this section, compression models are used to automatically break simple substitution ciphers for the Arabic language. A description of our method is given, followed by some results.

6.5.1 The Method

This section presents a full description of our new method for the automated decryption of Arabic simple substitution ciphertext. At the beginning, we have generated an Arabic simple substitution cipher in Unicode. Then, our new cryptanalysis approach is performed. The main idea of our approach, as mentioned in Chapter 3, is based on substituting one letter at a time throughout the text (starting with the most frequent), then one of the compression methods is used to calculate the codelength value using a similar technique to what was used for English in Chapter 3. PPMC, PPMD and Gzip compression schemes are used in our experiments.

The pseudo code of our approach is introduced in Algorithm 9. This is based on the same approach that was adopted for English. What is different here compared to that approach is that after removing all the non-alphabetical Arabic characters (excluding spaces) from the ciphertext in line 1, the text at this stage only comprises 37 Arabic letters. Also, the number of the best results to keep at each stage is set to be 1000 as shown in line 5. Experimental results concerning the Arabic language (as described below in the next section) show that keeping the 1000 best results in a buffer at each stage of the algorithm provides a good compromise. This compares with 500 best results that are kept for the English language.

Another important difference is that in order to calculate the codelength value for each Arabic permutation, a pre-processing operation is performed either using the Buckwalter Arabic transliteration 'BA-PPM' or an Arabic-specific encoding method using UTF-8 numbers 'CSA-PPM' (see line 13). (Since the results from Section 6.3 indicate that the compression ratio produced by both BA-PPM and CSA-PPM are exactly the same, we will keep on using the CSA-PPM method in our experiments). Since the Arabic language contains plenty of repeated bigraphs (as shown in Section 2.5.3), an adaptive bigraph substitution method for Arabic has also been investigated. Bigraph Substitution for PPM (BS-PPM) simply involves substituting the most frequent Arabic bigraphs with unique single symbols. Previous experiments have shown that substituting the top 100 bigraphs and expanding the alphabet by 100 more symbols produces better compression (Alhawiti, 2014). CSA-PPM and Gzip are character-based compression models and are adopted to calculate the compression codelength (lines 14 and 15) and BS-PPM uses a bigraph character-based method, and is adopted on lines 16 to 21. The final results are returned on line 29.

Algorithm 9: Pseudo code of the automatic cryptanalysis of Arabic simple substitution ciphers.

```
   Input  : ciphertext
   Output: decrypted text(s)
 1 remove all non-Arabic alphabet characters excluding spaces from the ciphertext;
 2 examine the ciphertext to create a sorted list of the zeroth order frequencies for the
   Arabic alphabet;
 3 replace the characters in the ciphertext with the special symbol '.';
 4 initialise Q1 (list) with a modified ciphertext;
 5 maximum size of Q2 (sorted list) ← 1000;
 6 foreach crypto character 'cc' in the zeroth order frequent characters (starting from the
   most to the least frequent characters) do
 7     Q2 ← empty;
 8     foreach ciphertext in Q1 do
 9         foreach alphabetic and space character 'ac' do
10             if 'ac' is not used before as a replacement of the previous crypto characters then
11                 replace each crypto character 'cc' in the ciphertext with the unused character 'ac'
                   as a candidate for 'cc';
12                 compute compression codelength:
13                     perform pre-processing operation by using Arabic-specific encoding method
                       on the ciphertext;
14                     if character-based compression method is performed then
15                         calculate codelength value of the preprocessed-ciphertext using the CSA-PPM
                           or Gzip compression model;
16                     else if bigraph character-based compression method is performed then
17                         divide the preprocessed-ciphertext into bigraphs;
18                         foreach bigraph in the preprocessed-ciphertext do
19                             if a bigraph within the 100 most frequent training text bigraphs then
20                                 replace the birgraph with the unique symbol;
21                         calculate codelength value of the preprocessed-ciphertext using the BS-PPM
                           compression model;
22                     return codelength value;
23                 if the size of the sorted list Q2 < 1000 then
24                     add the ciphertext to Q2;
25                 else if the codelength value of the last element in Q2 > codelength value of the
                   current ciphertext then
26                     remove the last element in Q2;
27                     add the ciphertext to Q2;
28     replace Q1 with Q2;
29 return Q1 containing the best solutions (the 'decrypted text(s)');
```

In Algorithm 10, a word-based PPM compression model is applied to the output produced from Algorithm 9. This model potentially provides a more accurate estimation of the quality of the solutions. The codelength values are re-calculated using this word-based model for Arabic.

Algorithm 10: Pseudo code of word-based ranking algorithm for the Arabic substitution.

```
Input  : the list Q1 (output from Algorithm 9)
Output: decrypted text(s)
1 maximum size of Q3 (sorted list) ← 1000;
2 foreach text in Q1 do
3     perform pre-processing operation by using Arabic-specific encoding method
      on the text;
4     calculate compression codelength value of the preprocessed-text using the
      CSA-PPM word-based compression method;
5     store the text in the sorted list Q3;
6 end
7 return Q3 containing the best solutions (the ‘decrypted text(s)’);
```

Table 6.4 presents the different variants that we have used in our experiments with the label assigned to each one. Order 5 CSA-PPM models have been employed. Experiments with a full range of variations have been conducted (CSA-PPMC, or CSA-PPMD, with and without update exclusions, with and without explicit order -1 codelengths; and Gzip). The specific codelength value we assign for order -1 contexts for the new variation of the PPM model is 28.90 ($= -\log_2 \frac{1}{N}$, where N is the size of the Arabic training data, $N = 513,291,607$).

6.5.2 Experimental Results

Experimental results for the different corpora and variants are discussed in this section. Different Arabic corpora types (classical, modern and mixed) with different sizes (small, medium and large) are used in our experiments to explore which is the most effective corpus to use in the automatic cryptanalysis of the Arabic language ciphers. These corpora were adopted to

Table 6.4: Some compression method variants used for the automatic cryptanalysis of Arabic simple substitution ciphers.

Name	**Compression method**
Variant A	CSA-PPM without update exclusions
Variant B	CSA-PPM without update exclusions with the same specific codelength value assigned to all order -1 context predictions
Variant C	Standard CSA-PPM (i.e with update exclusions)
Variant D	Standard CSA-PPM with the same specific codelength value assigned to all order -1 context predictions
Variant G	Gzip

train the PPM models using 37 character Arabic text. A corpus of 120 different Arabic ciphertexts chosen at random from many various resources with different lengths (ranging from 44 characters to almost 750 characters) are used as testing texts. Samples of which are listed in Appendix IV.

Table 6.5 presents the average number of errors for the different corpora used in our experiments. These corpora are used to train a new variation of the PPM algorithm where a specific codelength value is assigned when encoding all order -1 contexts (Variant B). This variant showed better performance than the others. The results indicate that in most cases the larger corpora produce a lower number of errors than the smaller corpora. The results also show that the Mixed corpora performs better than the other corpora type. In addition, the buffer size of 1000 provides a good compromise between the size of memory used and the number of errors produced.

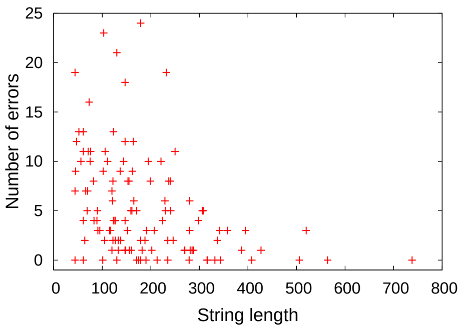
Table 6.6 presents results regarding the average number of errors for the 120 Arabic ciphertexts when using the 1000 buffer size. It is clear that the Large Mixed corpus performs better than the other corpora with an indication that the Large Mixed corpus is the best corpus to use in the automatic cryptanalysis of Arabic ciphers. Figures 6.2, 6.3 and 6.4 demonstrate the number of errors for the different corpora for different cryptogram lengths.

Table 6.5: Average number of errors for our different corpora with different categories and different buffer size.

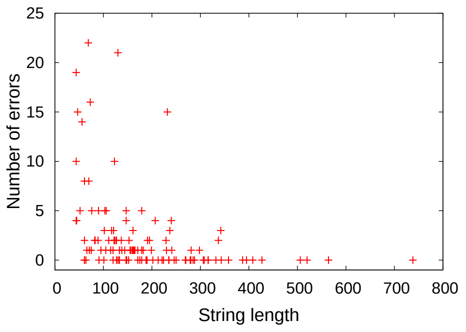
Corpus type	**Category**	**Buffer size**	**Texts without errors (out of 120)**	**Avg. # errors for best solution**	**Avg. # errors for best ten solutions**
Classical	Small	500	19	5.57	6.05
		1000	19	5.26	5.73
		2000	19	4.87	5.43
	Medium	500	50	2.60	3.30
		1000	50	2.40	3.08
		2000	51	2.07	2.71
	Large	500	49	2.78	3.53
		1000	49	2.28	3.03
		2000	49	2.13	2.81
Modern	Small	500	48	2.62	3.33
		1000	49	2.43	3.12
		2000	51	2.19	2.94
	Medium	500	54	2.42	3.27
		1000	57	2.12	2.97
		2000	57	2.01	2.81
	Large	500	59	2.15	3.11
		1000	62	1.91	2.90
		2000	63	1.67	2.64
Mixed	Small	500	60	1.98	2.77
		1000	62	1.54	2.34
		2000	62	1.51	2.30
	Medium	500	61	1.78	2.69
		1000	62	1.60	2.44
		2000	62	1.42	2.37
	Large	500	63	1.53	2.34
		1000	66	1.14	2.05
		2000	66	1.14	2.01

Table 6.6: Average number of errors for the different corpora with different categories when examining the best solution and a 1000 buffer size.

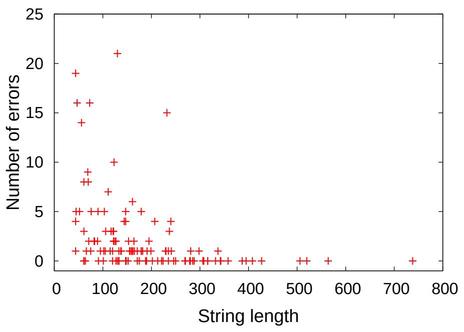
Category	**Classical corpora**	**Modern corpora**	**Mixed corpora**
Small	5.26	2.43	1.54
Medium	2.40	2.12	1.60
Large	2.28	1.91	**1.14**



(a) Small size



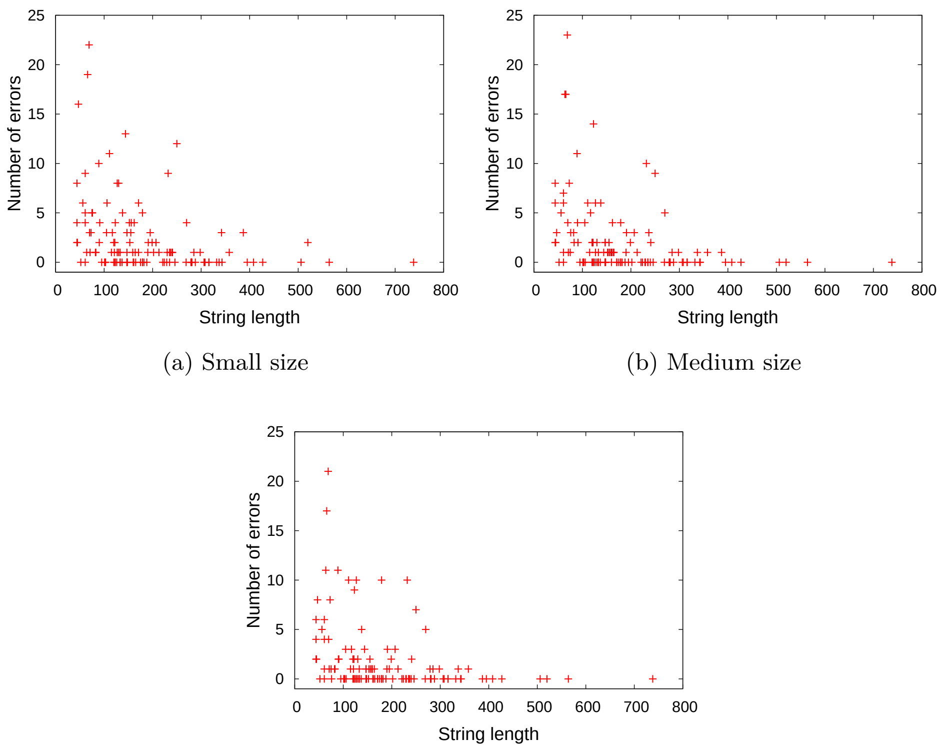
(b) Medium size



(c) Large size

Figure 6.2: Errors produced from different classical corpora when using buffer size of 1000 for variant B.

As stated, various 120 Arabic ciphertexts with different lengths have been examined in our experiments. In order to measure the success and the accuracy of our automatic cryptanalysis algorithms, alphabetic substitution errors (mapping errors) are counted. Referring to the different variants described in Table 6.4 and by using the large Mixed corpus as a training text, the results showed that only when using the new CSA-PPM variant (Variant B) as a method of calculating the codelength values resulted in successfully decrypted texts. On the other hand, the other CSA-PPM variants produced a significantly greater number of errors. The same was repeated when using the Gzip algorithm in the last variant (G). Arabic example output produced by the different variants is shown in Table 6.7.


(a) Small size

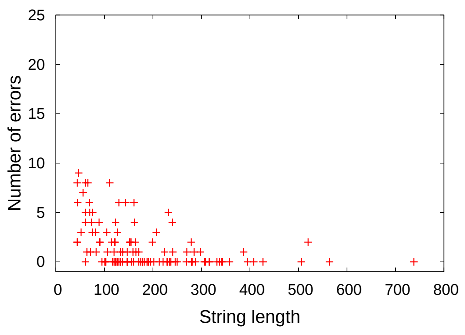
(b) Medium size

(c) Large size

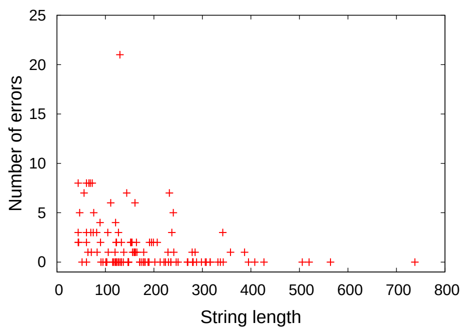
Figure 6.3: Errors produced from different Modern corpora when using buffer size of 1000 for variant B.

Table 6.7: Sample of solved Arabic simple substitution ciphertexts by different variants.

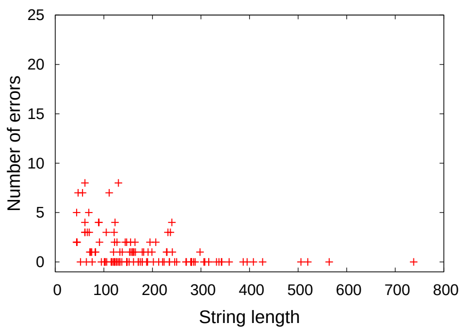
Variants	Number of errors	Decrypted message
Variant A	15	القماشوينزلقولهؤيدوهروالاجنةهامانو البرناتو الخادمان
Variant B	0	العراق يتطلع لمزيد من الاستثمارات والإنتاج والصادرات
Variant C	17	لأرإلكفتوأرفأميت فمغفلاألشىءملإلنفالأطغلهفالأصل إلى
Variant D	19	هرأنهحظة ؤرأظريةعظيمظهرهف ليهنه ظاهركم هوظاهرتهعنه
Variant G	19	نهأسنحظر عهأظهيبرمظية ظنهنؤ لينسن ظفنهك نوظفنهتنمسن



(a) Small size



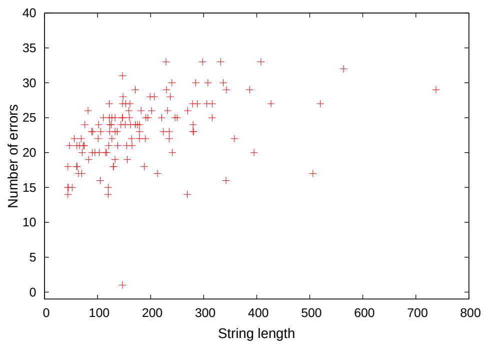
(b) Medium size



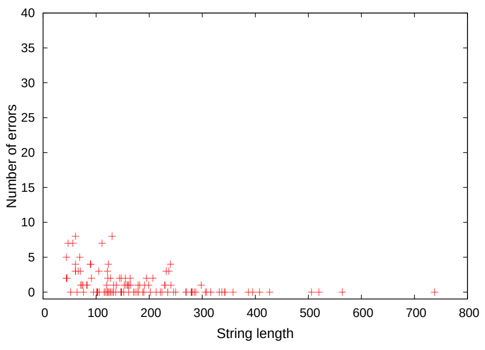
(c) Large size

Figure 6.4: Errors produced from different Mixed corpora when using buffer size of 1000 for variant B.

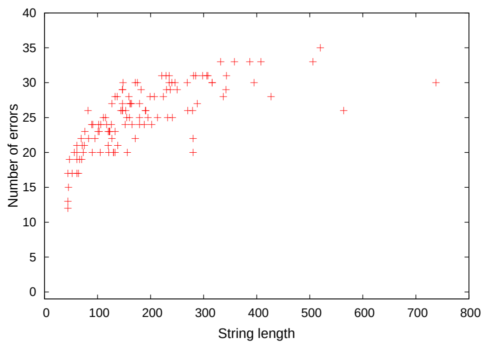
Figure 6.5 illustrates the number of errors for each testing ciphertext for the different variants (A to G) as a result of our cryptanalysis method. It is clear that the number of errors for all the variants, except Variant B, is very high. In Variant B, a new variation of CSA-PPM without update exclusions with the same specific codelength value assigned to all order -1 context predictions is used. The results show that 55% of the ciphertexts are correctly solved with no errors (that is, the best solution with minimum codelength value is the correct solution). Also, a further 28% of the examples are decrypted with just one or two errors. All the cryptograms of length longer than 300 are successfully solved without any errors.



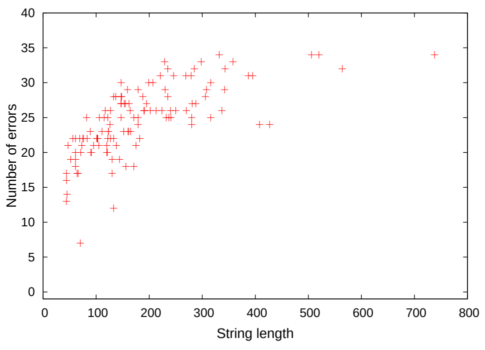
(a) Errors produced from variant A.



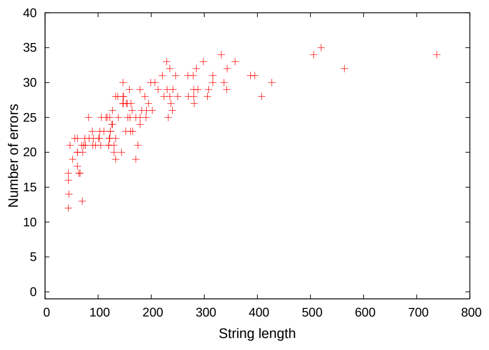
(b) Errors produced from variant B.



(c) Errors produced from variant C.



(d) Errors produced from variant D.



(e) Errors produced from variant G.

Figure 6.5: Errors produced from different Arabic variants.

The average number of automatic cryptanalysis errors for each variant for the 120 cryptograms using the CSA-PPMD models is presented in tables 6.8a and 6.8b. Clearly, the best performing variant overall is Variant B. Variant

G came up with the worst result with a high average number of errors. Note that the CSA-PPMC models produced slightly worse results than the CSA-PPMD models.

Table 6.8a: Average number of errors for each different Arabic variant when examining the best solution.

Variants	**A**	**B**	**C**	**D**	**G**
Average errors	23.10	1.14	25.18	24.63	25.38

Table 6.8b: Average number of errors for each different Arabic variant when examining the ten best solutions.

Variants	**A**	**B**	**C**	**D**	**G**
Average errors	23.11	2.05	25.19	24.68	25.41

Irvine (1997) stated that a better language model would not help on shorter cryptograms because they are inherently ambiguous. This is because the unicity distance for an Arabic simple substitution is about 44, since the entropy of the order-5 Arabic model is $H_5 = 1.923$ (see Section 6.3),

$$U = \frac{\log 37!}{\log 37 - H_5} = 43.6 \ .$$

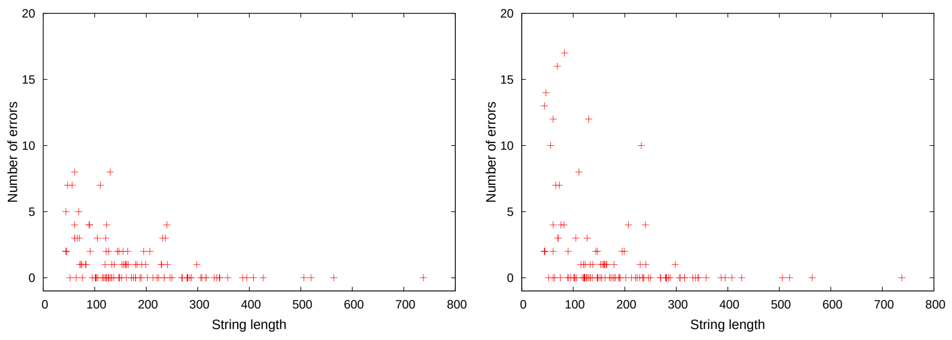
So, according to Irvine we cannot consistently expect to solve Arabic cryptograms shorter than this number of characters. We have confirmed this in our experiments here, where we have found that various short cryptograms of 44 characters or so can be effectively solved but not less. On the other hand, this is not the case for the English language in Chapter 3, where many different short cryptograms, shorter than the unicity distance, were effectively decrypted using our approach.

As stated earlier in Chapter 2, the occurrence of many bigraphs is much more than some single characters in the Arabic language. Thus, we also investigate the use of Bigraph Substitution method for Arabic (BS-PPM) to explore if this results in better cryptanalysis. Results in Table 6.9 and

Figure 6.6 show that the number of ciphertexts that are correctly decrypted without errors has increased to reach 62.5% by using the bigraph-based method. However, the results also indicate that eight of the decrypted texts showed a very high number of errors.

Table 6.9: Average number of errors for the character-based and bigraph-based models for Variant B.

Models	Text without errors	Average best solution errors	Average best 10 solutions errors
Character-based	66	1.14	2.05
Bigraph-based (BS-PPM)	75	1.60	2.50



(a) Character-based model errors. (b) Bigraph-based model (BS-PPM) errors.

Figure 6.6: Errors produced from character-based and bigraph-based models for Variant B.

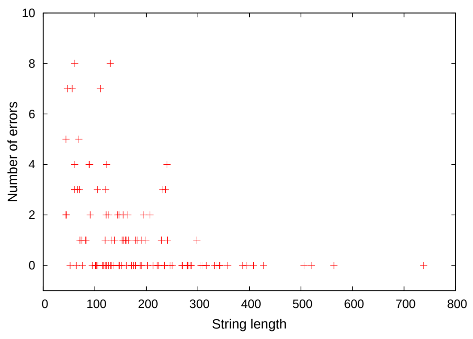
6.5.2.1 Improving Results using a Word-based PPM Compression Method for Arabic

Further investigation concerning how to improve our cryptanalysis method is introduced in this section. A word-based model combined with the character-based method led to further improved results for the English language experiments in Chapter 3. The effectiveness of applying this model as a secondary post-processing stage is also examined here to see if this also results in better cryptanalysis for Arabic.

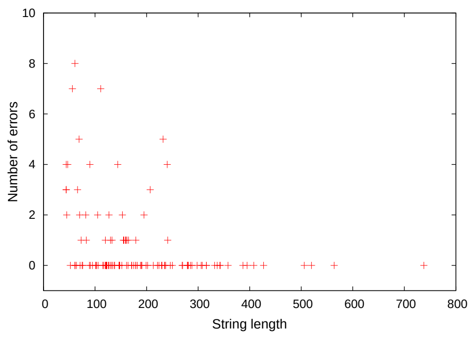
The average number of errors for both character-based and word-based models is shown in Table 6.10. Results in Table 6.10 and Figure 6.7 indicate that the secondary word-based method showed better cryptanalysis performance than the character-based model alone in Variant B when the order 5 CSA-PPMD model is used. After applying this word-based method, about 72% of cryptograms are now solved without any errors, with an improvement of 17%. The results in Table 6.10 also show how the word-based approach improves the average number of errors for the best solutions and for the ten best solutions.

Table 6.10: Average number of errors for the character-based and word-based models.

Models	Text without errors	Average best solution errors	Average best 10 solutions errors
Character-based	66	1.14	2.05
Character-based&word-based	86	0.76	1.99



(a) Character-based model errors.



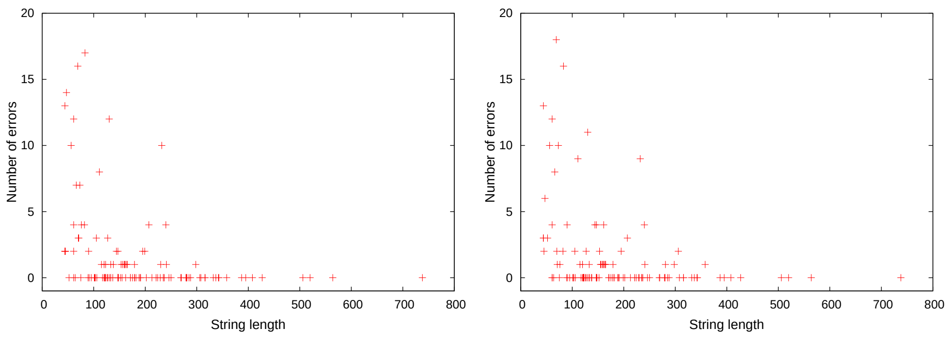
(b) Character-based model & word-based model errors.

Figure 6.7: Errors produced from the character-based and word-based models for Variant B.

Results regarding the bigraph-based and word-based models are presented in Table 6.11 and Figure 6.8. It is clear that when the bigraph-based method is combined with the word-based method, it did not lead to much further improved results.

Table 6.11: Average number of errors for the bigragh-based and word-based models.

Models	Text without errors	Average best solution errors	Average best 10 solutions errors
Bigraph-based	75	1.60	2.50
Bigraph-based&word-based	74	1.59	2.48



(a) Bigraph-based model errors. (b) Bigraph-based model & word-based model errors.

Figure 6.8: Errors produced from the bigraph-based and word-based models for Variant B.

A comparison between the different Arabic models when examining the best solution is showed in Table 6.12. Clearly, we can see that the lowest average number of errors is obtained from an efficient combination between the character-based model (Variant B) and the word-based model in column three.

Table 6.12: Average errors for the different Arabic methods.

Models	**Character**	**Bigraph**	**Char.&word-based**	**Bigraph&word-based**
Avg. errors	1.14	1.60	0.76	1.59

The results also showed that the time needed to break Arabic ciphertexts is almost double the time required to break English cryptograms as presented in Table 6.13. This is because the size of the buffer of the best results that are kept at each stage for the Arabic language is twice that of English. As well, the extra steps that are needed to perform the preprocessing operation.

Table 6.13: Average time required to automatically break different Arabic simple substitution ciphertexts.

Cipher Length	**50**	**100**	**200**	**400**
Time *(Sec)*	6.03	7.12	10.56	15.23

The summary of results regarding our best performing character-based method (Variant B) and when it is combined with the word-based method is presented in Table 6.14. The results clearly show that the best cryptanalysis performance was achieved by using the character-based PPM method followed by the word-based PPM method. About 72% of the Arabic cryptograms are successfully broken without any errors and over 91% are decrypted with just three errors or less.

Table 6.14: Summary of results.

Errors	**Character-based model**		**Character & word-based models**	
	No. of ciphertexts	Cumulative percentage (%)	No. of ciphertexts	Cumulative percentage (%)
0	66	55.00	86	71.67
≤ 1	88	73.33	99	82.50
≤ 2	100	83.33	106	88.33
≤ 3	108	90.00	110	91.67
≤ 4	113	94.17	115	95.83
≤ 5	115	95.83	117	97.50
≤ 6	115	95.83	117	97.50
≤ 7	118	98.33	119	99.17
≤ 8	120	100.00	120	100.00

These results indicate excellent performance with Arabic language on substitution ciphers. The next section examines transposition ciphers.

6.6 An Automatic Cryptanalysis of Arabic Transposition Ciphers

This section now reviews Arabic transposition ciphers and the automatic cryptanalysis of this cipher specifically. In a transposition cipher, the content of a message is obscured by rearranging groups of symbols; a transposition is thus a permutation. Originally, matrices were used. For example, encoding "أنظمةالتشفير", which means "encryption systems", using a 3×4 matrix, Figure 6.9, produces "أةشنافظليمتر" if the columns are read off in order.

م	ظ	ن	أ
ت	ل	ا	ة
ر	ي	ف	ش

Figure 6.9: Example of a matrix transposition cipher.

By reading the columns in a different order, say 3–2–1–4, different ciphertexts can be obtained, such as "ظلينافأةشمتر". The technique can be extended to *d* dimensions. In transposition ciphers, a plaintext block is encrypted into a ciphertext block using a fixed permutation (Giddy and Safavi-Naini, 1994). For example, if

$$f(x) = \begin{cases} 3 & \text{if } x = 1 \\ 2 & \text{if } x = 2 \\ 4 & \text{if } x = 3 \\ 1 & \text{if } x = 4 \end{cases}$$

then, "أنظم ةالت شفير" would be encoded as "مناأظ تاةل رقشي".

Our cryptanalysis method regarding Arabic transposition ciphers is described in detail in the next subsection, followed by the experimental results that were produced.

6.6.1 The Method

As with the previous solution as detailed in Section 6.5.1, a compression scheme is also used to calculate the codelength value to use as a metric for ranking the quality of each possible permutation.

First, we have generated an Arabic transposition cipher in Unicode, then the cryptanalysis method is performed. Our new cryptanalysis method consists of two main phases, the same as for the English language experiment. The first phase (Phase I) is based on trying to automatically crack an Arabic ciphertext using a transposition of specified size by exhaustively computing all possible transpositions, while the second phase (Phase II) is based on achieving readability.

The pseudo code for the first part of our approach is presented in Algorithm 11. This algorithm is similar to what was used for English. What is different is that the text now consists of only 36 Arabic letters after removing all the non-alphabetical Arabic characters (including spaces) from the ciphertext as presented in line 1. Also, in order to calculate the codelength value for each permutation, a pre-processing operation using an Arabic-specific encoding method is performed in line 4. Then, all possible transpositions are generated, and the compression codelength recomputed at each stage (lines 5 to 14).

The second phase of our approach, as mentioned before, focuses on achieving readability. The second assessment step is performed in this phase through adding spaces automatically to the decrypted messages that was produced from Phase I, and then assessing the quality of the solutions by computing the codelength values for each possible message. As we did for the English language experiment, the two alternative ways of achieving readability have also been investigated here for the Arabic language. In the first method, Phase II-A, according to the decrypted message permutation which is outputted from the previous phase, all further possibilities are explored where a space is inserted after each character. Underperforming possibili-

Algorithm 11: Pseudo code of the main decryption phase (Phase I) for the Arabic transposition.

```
Input   : ciphertext
Output: decrypted text
1  remove all non-Arabic alphabet characters and spaces from the ciphertext;
2  maximum size of Q (priority queue) ← 3;
3  maximum key-size of transposition ← 12;
4  perform pre-processing operation by using Arabic-specific encoding method on
   the ciphertext;
5  foreach key-size do
6      if ciphertext-length mod Key-size = 0 then
7          divide the ciphertext into blocks according to key-size;
8          perform a permutation over each ciphertext blocks;
9          foreach possible permutation do
10             calculate codelength value using the CSA-PPM or Gzip compression
               model;
11             store a permutation with smaller codelength value in Q;
12         end
13     end
14 end
15 return the priority queue 'Q' (the 'decrypted text');
```

ties which have bad codelength values are pruned from the priority queue and only the best performing possibilities are returned at the end. In the second method, Phase II-B, the Viterbi algorithm is used to search for the best probable segmentation sequences. In both these two methods, a post-processing operation is performed afterwards, by using the Arabic-specific decoding method.

Two Arabic adapted variants of the PPM modelling system, CSA-PPMD and CSA-PPMC, have been used in our method. Also two forms of these schemes are examined, one with update exclusions (i.e the standard CSA-PPMD and CSA-PPMC) and one without update exclusions. The use of the Gzip compression scheme is examined as well, in order to explore the most effective method that can be applied to the problem of automatically recognizing the valid decryption. Only five variations are shown in Table 6.15 either to illustrate the best performing schemes or to illustrate interesting results for comparison.

Table 6.15: Arabic variants used in the automatic cryptanalysis of Arabic transposition ciphers.

Variants	**Phase I**	**Phase II-A**	**Phase II-B**
A	CSA-PPM without update exclusions	CSA-PPM without update exclusions	
B	CSA-PPM without update exclusions		CSA-PPM without update exclusions
C	CSA-PPM with update exclusions	CSA-PPM with update exclusions	
D	CSA-PPM with update exclusions		CSA-PPM with update exclusions
G	Gzip	Gzip	

6.6.2 Experimental Results

The experimental results are discussed in this section. In our experiments, the order-5 CSA-PPM models have been trained on a corpus of many different Arabic books and novels from different topics using 36 and 37 Arabic character (when space is included). As explained above, classical and modern Arabic texts are used to produce this large Mixed Arabic corpus. Regarding the cryptograms test corpus, 125 different Arabic cryptograms were chosen at random from different resources to be used as testing texts. Cryptogram lengths ranged from 17 to almost 750 letters (samples of which are in Appendix V).

A sample of decryption is shown in Table 6.16 for the Arabic cryptogram ‘ميكقنصدن دكعوكهاو اأرغخمنك’. Compression codelengths are listed in bits with the lowest three results presented for the different phases.

We have created a transposition cipher for the Arabic language that depends on using the Unicode encoding. For each run, a random key is

Table 6.16: Output produced for the sample cryptogram ‘ميكقنصدن دكعوكهاو أأرغخمنك’.

Phase I	**Phase II-A**	**Phase II-B**
Variants A and B		
108.64 صديقكمننهاكوعدوكمناأغراك 110.62 نكمنصديقكعدوهاكوراكمناأغ 111.37 يكمننصدقكعدوكهاوأراكمنغ	111.03 صديقك من نهاك وعدوك من أغراك 116.22 نك من صديقك عدوها كورا كمن أغ 121.21 يك من نصدقك عدوكها وأراك منغ	صديقك من نهاك وعدوك من أغراك يك من نصدقك عدوكها وأراك منغ نك من صديقك عدوها كورا كمن أغ
Variants C and D		
108.15 منكنصديقدوعكهاكواكرمناأغ 110.76 قكمنصدينوعدوهاككغراكمناً 110.85 صديقكمننهاكوعدوكمناأغراك	112.38 صديقك من نهاك وعدوك من أغراك 122.53 قك من صدين وعدوها ككغراكمناً 124.41 منكن صديق دوعك هاك واكر من أغ	صديقك من نهاك وعدوك من أغراك قك من صدين وعدوها ككغراكمناً منكن صديق دوعك هاك واكر من أغ

generated to encrypt the original message (plaintext). Then, we have applied our cryptanalysis method to automatically break this message (ciphertext). Different permutation key sizes (from 2 to 12) and different cryptograms with different lengths have been examined in our experiments.

In order to measure the differences between the original texts and the decrypted texts that resulted from our decryption method, the Levenshtein distance metric is used (Levenshtein, 1966). The experimental results of Phase I show that for the first two variants A and B, when the CSA-PPM method without update exclusions is used, 97% of the cryptograms are successfully decrypted without any errors. For the other two variants C and D, when the standard CSA-PPM method is used, the results showed that 95% of the ciphertext are solved with no errors. For the last variant, Variant G, by using the Gzip scheme, a success rate of 90% was achieved.

The results of Phase II show that for the two variants A and B, 97% of the cryptograms are successfully decrypted and segmented. For the other variants (C and D), which resulted in a 95% decryption success rate from Phase I, now 97% of the ciphertexts are effectively decrypted and segmented as a result of Phase II.

It is important to note that the second phase of our approach has the ability to find better solutions not found by the first phase. For example, referring to Table 6.16, the best two solutions as output from using Variants

C and D shown in column one are not correct (the third solution is the correct one). However, after the second phase has been applied, the best segmentation of the third solution in column one now appears as the best solution in columns two and three which is correct.

Arabic word segmentation is considered a difficult problem (Zeki, 2005). As the Arabic language is a rich morphological language, this characteristic represents a real challenge for identifying the correct segmentation among many possible corrected alternatives. For example, the word " ورده" could be segmented into " ورده " and " و رده", which means "rose" and "and his reply". Both of these segmented forms are correct, but with completely different meanings and contexts. However, according to our approach and results, in almost all cases the proper readable solutions were obtained for all different variants, except for the last variant (Variant G).

Output examples (with spaces) produced from different variants are exhibited in Table 6.17. For example, this table shows how 'إناللهلاينظرإلىصوركموأموالكمولكنينظرإلىقلوبكموأعمالكم', which is the best result produced from Phase I for a sample cryptogram, is effectively segmented with no errors using both variants B and C, one and two errors using variants A and C, while Variant G shows poor performance with nine segmentation errors.

Figure 6.10 presents scatter-plots for the number of errors (on the *y* axis) versus the string lengths of cryptograms (on the *x* axis) for the different variants when the PPMD models are used. For example, Figure 6.10a presents the number of word segmentation errors for each decrypted message that was output from Variant A. This figure shows that most of the errors are less than four. About half of the cryptograms are correctly segmented without any errors and a third with four errors or less. However, three of the cryptograms suffered from more than ten errors, and one cryptogram of length 400 letters produced 23 errors.

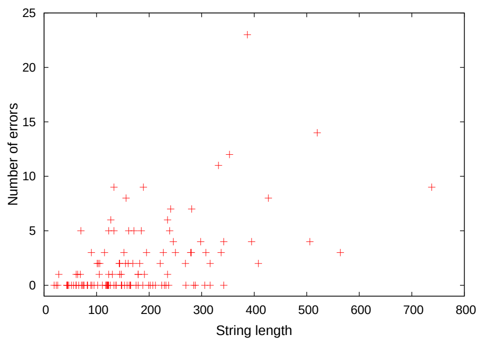
Variant B offered better performance than the other variants. More than 63% of the testing texts have been correctly segmented with no errors and a further quarter of them with three errors or less. Just four cryptograms

Table 6.17: Example of solved Arabic transposition ciphertexts with spaces inserted by the five different variants.

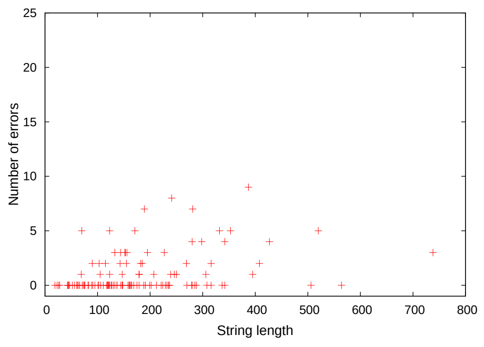
Variants	**Number of errors**	**Decrypted message (with spaces)**
Variant A	1	إن الله لا ينظر إلى صوركم وأموالكم ولكن ينظر إلى قلوبكم وأعمالك م
Variant B	0	إن الله لا ينظر إلى صوركم وأموالكم ولكن ينظر إلى قلوبكم وأعمالكم
Variant C	2	إن الله لا ينظر إلى صوركم وأموالكم ولكن ينظر إلى قلوبك م وأعمالك م
Variant D	0	إن الله لا ينظر إلى صوركم وأموالكم ولكن ينظر إلى قلوبكم وأعمالكم
Variant G	9	إن اللهلاينظرإلىصوركموأموالكم ولكنينظرإلىقلوبكموأعمالكم

ended up with seven, eight or nine errors. Figure 6.10b exhibits this variant's results. Variant C showed similar results to Variant A with slight differences as shown in Figure 6.10c. On the other hand, Variant D showed a slightly worse result than Variant B. Fifty three percent of the cryptograms were effectively segmented with no errors and forty percent were segmented with four errors or less. Only one cryptogram consisting of 353 letters ended with more errors (ten). Figure 6.10d outlines Variant D results. The number of errors produced from Variant G is shown in Figure 6.10e. It is clear that the number of errors for each decrypted ciphertext in this case is much higher, with most of the errors being greater than 20. Also none of the cryptograms finished with no errors.

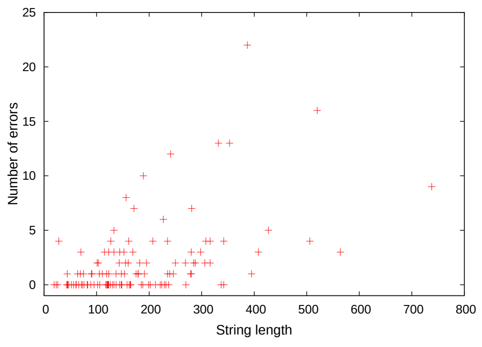
Table 6.18 presents results concerning the average number of space insertion errors for the 125 texts we experimented with for each of the variants. The CSA-PPMD model is used for the first four variants. It is clear that the average number of errors for all the variants, except Variant G, is quite low with Variant B performing best. Slightly worse results were obtained when the CSA-PPMC models are used.



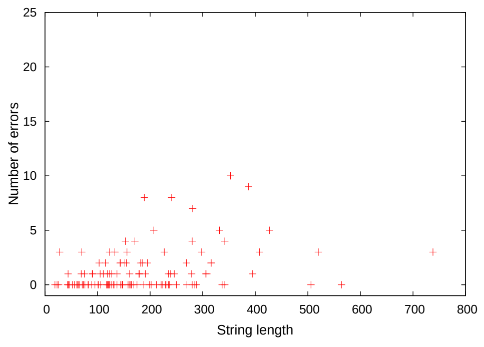
(a) Errors produced from Variant A.



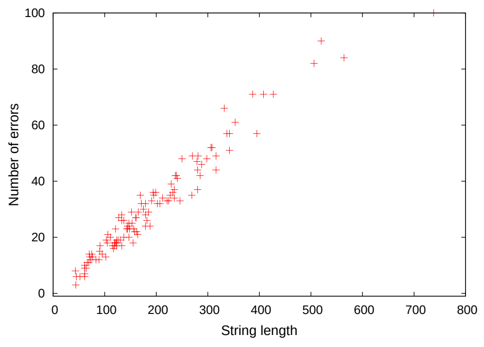
(b) Errors produced from Variant B.



(c) Errors produced from Variant C.



(d) Errors produced from Variant D.



(e) Errors produced from Variant G.

Figure 6.10: Segmentation errors produced from the five different variants for the Arabic transposition.

Table 6.18: Average number of errors for the Phase-II variants.

Variants	**A**	**B**	**C**	**D**	**G**
Average errors	2.09	1.10	2.03	1.23	30.43

To investigate more about the accuracy of the compression methods that we used in segmenting the 125 Arabic decrypted texts, three main metrics are used: recall rate, precision rate and error rate. The results presented in Table 6.19 indicate that the variants which depend on the CSA-PPMD compression methods were able to achieve very high recall and precision rates, reflecting the quality of the Arabic word segmentation. The error rates resulted from unusual words not included in the training text.

Table 6.19: Evaluating the quality of the Arabic word segmentation.

Variants	**Recall rate%**	**Precision rate%**	**Error rate%**
Variant A	93.38	93.52	6.62
Variant B	96.35	95.94	3.65
Variant C	93.08	94.15	6.92
Variant D	95.50	95.67	4.50
Variant G	1.53	10.02	98.47

The average time that is needed to recognize the correct solutions for the different Arabic transposition cryptograms with different block sizes is shown in Table 6.20. This table presents the average elapsed time for the automatic cryptanalysis of three different length cryptograms, for both Phase I and Phase II. For each cryptogram, the average elapsed time in seconds is shown after running the approach ten times. The results show that the average time increases as the key size increases but overall the method is reasonably efficient.

In summary, experimental results showed that for the first phase 'Phase I', a successful decryption rate of 97% was achieved when the CSA-PPM method without update exclusions was used and a 95% when the standard CSA-PPM model (with update exclusions) was used. Two methods have been used in the second phase in order to achieve readability. The first one is based on using a priority queue and the second depends on the Viterbi algorithm. A successful decryption rate of 97% was achieved using CSA-PPM models with the use or not of update exclusions as a result of this phase. In

Table 6.20: Average time required (across 10 runs) to automatically break Arabic transposition cryptograms with different lengths and different keys size.

Ciphertext	**Key size**					
length	**Time** (in seconds)					
(Letters)	**5**	**6**	**7**	**8**	**9**	**10**
40	0.85	0.87	0.90	1.08	2.60	12.30
150	1.23	1.27	1.42	1.90	13.93	48.20
300	3.54	3.77	3.89	4.94	23.13	95.50

addition, almost all the decrypted examples were effectively segmented with a quite low average number of errors. Marginally better results are gained from using the Viterbi algorithm but with a slightly slower execution time (almost one error on average as a result of Phase II-B, and slightly greater than two errors on average for Phase II-A). In comparison with the English language, slightly worst results were obtained for the Arabic language (the average number of errors for the English texts is less than one). The best findings resulted from using Variant B, when the CSA-PPM model without update exclusions with the Viterbi method is performed.

The next section examines Playfair ciphers to further confirm whether the efficient performance with Arabic language on substitution and transposition ciphers is also reproduced for another cryptosystem.

6.7 An Automatic Cryptanalysis of Arabic Playfair Ciphers

For Playfair ciphers, the English alphabet is arranged in a 5×5 key matrix based on secret key. For Arabic, 36 alphabetic letters are arranged into a 6×6 grid. For example, if the keyword 'التشفير' is used, the key grid would be as below:

To encrypt the plaintext message "القناعة كنز لا يفنى", the encrypted mes-

ا	ل	ت	ش	ف	ي
ر	ب	ث	ج	ح	خ
د	ذ	ز	س	ص	ض
ط	ظ	ع	غ	ق	ك
م	ن	ه	و	أ	إ
آ	ى	ء	ؤ	ئ	ة

sage would be: "لت ظأ تط يإ هذ تل اي بل".

If any bigrams contain repeated letters, an 'ؤ' letter is used to separate them. If the plaintext has an odd number of letters, an 'ؤ' is also inserted at the end so that the last letter is in a bigram.

The automatic cryptanalysis method of the Arabic Playfair cipher is presented in the next subsection. Experimental results are then discussed in the subsection that follows.

6.7.1 The Method

Using a similar approach as detailed in Chapter 5, the decryption method consists of two main phases. Phase I tries to automatically crack an Arabic Playfair ciphertext using a combination of the PPM compression method and a simulated annealing approach while Phase II tries to achieve readability. A variation of an order 5 CSA-PPMD model without update exclusions has been used in the experiments for both Phase I and Phase II.

Algorithm 12 presents the pseudo code for the first part of the method. This algorithm is similar to that for English in Algorithm 6 but differs in a number of respects. First of all, in a preprocessing step prior to the application of this algorithm, all non Arabic letters including spaces were omitted from the ciphertext (the text now consists of 36 Arabic letters). Then, in order to calculate the codelength value for each permutation, a pre-processing operation using an Arabic-specific encoding method is performed in line 1. The text is compressed using CSA-PPMD with an order 5 character-based model and the codelength value is produced. Also, for each temperature,

Algorithm 12: Pseudo code of Phase I for the Arabic Playfair.

```
Input  : ciphertext, Playfair grid-width
Output: deciphered-text
1  perform pre-processing operation by using Arabic-specific encoding method on the
   ciphertext
2  generate a random key
3  $currentBestKey \leftarrow randomKey$
4  decipher the ciphertext using the currentBestKey and calculate the codelength value
   using the CSA-PPM compression method
5  $currentBestScore \leftarrow -$ CSA-PPM-codelength score (decipher-text)
6  for $Iteration \leftarrow 0$ to 99 by 1 do
7      $maxKey \leftarrow currentBestKey$
8      decipher and calculate the codelength value using the CSA-PPM compression
       method
9      $maxScore \leftarrow -$ CSA-PPM-codelength score (decipher-text)
10     for $Temp \leftarrow 20$ downto 0 by 0.2 do
11         for $Count \leftarrow 0$ to 19999 by 1 do
12             modify maxKey by choose a random number between (1,50):
13                 if the number is 0 then swap two rows, chosen at random
14                 if the number is 1 then swap two columns, chosen at random
15                 if the number is 2 then reverse the key
16                 if the number is 3 then reflect the key vertically, flip top to bottom
17                 if the number is 4 then reflect the key horizontally, flip left to right
18                 if any other number then swap two characters at random
19             $newKey \leftarrow modified\text{-}maxKey$
20             decipher and calculate the codelength value using the CSA-PPM compression
               method
21             $newScore \leftarrow -$ CSA-PPM-codelength score (decipher-text)
22             calculate $diff \leftarrow newScore - maxScore$
23             if $diff >= 0$ then $\{maxScore \leftarrow newScore; maxKey \leftarrow newKey\}$
24             else if $Temp > 0$ then
25                 calculate $probability \leftarrow \exp(diff/Temp)$
26                 generate a random number between $(0,1)$
27                 if $probability > randomNumber$ then
28                     $\{maxScore \leftarrow newScore; maxKey \leftarrow newKey\}$
29             if $maxScore > currentBestScore$ then
30                 $currentBestScore \leftarrow maxScore; currentBestKey \leftarrow maxKey$
31                 Make systematic rearrangements(ciphertext, currentBestKey,
                   currentBestScore)
32         end
33     end
34 end
35 return the deciphered text with the best key
```

20,000 keys are tested instead of 10,000 keys used for English (line 11). Algorithm 7 then is called to perform some systematic rearrangements (see line 31). Finally, the deciphered text is returned with the smallest codelength value which represents the best solution found (line 35).

It is worth noting that negative scores based on the CSA-PPMD codelengths values are also used here in this algorithm, in order to maximize rather than minimize scores for the simulated annealing process.

The second phase of our approach, as stated before, is focused on achieving readability. The Viterbi algorithm is used in this phase, afterwards a post-processing operation is performed, by using the Arabic-specific decoding method.

6.7.2 Experimental Results

In this section, the experimental results of the new approach are discussed. An order 5 CSA-PPMD model has been trained on the large Mixed Arabic corpus using 36 Arabic letters. Seventy different Arabic cryptograms (samples of which are in Appendix VI) were randomly chosen from different resources as testing texts. These cryptograms range from 100 to 500 letters.

Our first example, Figure 6.11, illustrates the robustness of the compression based approach. The ciphertext is a 122 letter quote by 'Amani Alenzi': "لا وقت للذكريات ولا الماضي، العالم لم يتذكرنا بعد وسينسانا بالرغم من ذلك، سنظل معلقين هنا في هذه الحياة متشبثين بالامنا واحلامنا وانكسارتنا الخاصة".

The result shows how our new method is able to solve this short Arabic cryptogram without any errors. The execution trace is shown in the figure with the best score as it changes during the execution of the Phase I algorithm. The scores are increasing (i.e the codelengths are decreasing).

```
Iteration:          75
Mutation gain:      -471.74   لاةكتللذكرياتولاالراضيالعالمليتذكرنابعدةعينقانابالرغممنذلكقوملمعلحينهنافيهذهالصي
                              أامتشبثينبالامناواصلامناوانكقامتنارالخاءأ
                              Key: جصخربثةئأآسؤوإمنهقءضدذزغحكطظعشفيالت
Mutation gain:      -468.82   لاجقتللذكرياتولاالراضيالعالمليتذكرنابعدأعينقانابالرغممنذلكقوملمعلحينهنافيهذهالصي
                              اةمتشبثينبالامناواصلامناوانكقامتنارالخاءة
                              Key: جصخربثأئةآسؤوإمنهقءضدذزغحكطظعشفيالت
Mutation gain:      -466.58   لاجقتللذكرياتولاالراضيالعالمليتذكرنابعدأعينقانابالرغممنذلكقوملمعلحينهنافيهذهالسي
                              اةمتشبثينبالامناواسلامناوانكقامتنارالخاءة
                              Key: جسخربثأئةآصؤوإمنهقءضدذزغحكطظعشفيالت
Mutation gain:      -458.96   لاجقتللذكرياتولاالراضيالعالمليتذكرنابعدأعينقانابالرغممنذلكقوملمعلسينهنافيهذهالحي
                              اةمتشبثينبالامناواحلامناوانكقامتنارالخاءة
                              Key: جحخربثأئةآصؤوإمنهقءضدذزغسكطظعشفيالت
Mutation gain:      -443.95   لاجقتللذكرياتولاالراضيالعالمليتذكرنابعدأعينقانابالرغممنذلكقوملمعلسينهنافيهذهالحي
                              اةمتشبثينبالامناواحلامناوانكقامتنارالخاصة
                              Key: جحخربثأئةآءؤوإمنهقصضدذزغسكطظعشفيالت
Row swap gain:      -433.97   لاوقتللذكرياتولاالماضيالعالمليتذكرنابعدأعينقانابالرغممنذلكقوملمعلسينهنافيهذهالحي
                              اةمتشبثينبالامناواحلامناوانكقاآتنامالخاصة
                              Key: جحخربثوإمنهأئةآءؤقصضدذزغسكطظعشفيالت
Row swap gain:      -426.53   لاوقتللذكرياتولاالماضيالعالمليتذكرنابعدأعينقانابالرغممنذلكقوملمعلسينهنافيهذهالحي
                              اةمتشبثينبالامناواحلامناوانكقارتنامالخاصة
                              Key: جحخربثقصضدذزغسكطظعوإمنهأئةآءؤشفيالت
2-Mutation gain:    -423.31   لاوقتللذكرياتولاالماضيالعالمليتذكرنابعدىعينسانابالرغممنذلكسءولمعلقينهنافيهذهالح
                              ياةمتشبثينبالامناواحلامناوانكسارتنامالخاصة
                              Key: جحخربسصضدذزغقكطظعوأإمنهئةآءؤشفيالت
Mutation gain:      -416.59   لاوقتللذكرياتولاالماضيالعالمليتذكرنابعدءعينسانابالرغممنذلكسنظلمعلقينهنافيهذهالح
                              ياةمتشبثينبالامناواحلامناوانكسارتنامالخاصة
                              Key: جحخربسصضدذزغقكطظعوأإمنهءئةآؤشفيالت
Mutation gain:      -415.48   لاءعتللذكرياتولاالماضيالعالمليتذكرنابعدوقينسانابالرغممنذلكسنظلمعلقينهنافيهذهالح
                              ياةمتشبثينبالامناواحلامناوانكسارتنامالخاصة
                              Key: جحخربسصضدذزغقكطظعوؤإمنهءئةآأشفيالت
Mutation gain:      -414.25   لاوستللذكرياتولاالماضيالعالمليتذكرنابعدوقينسانابالرغممنذلكسنظلمعلقينهنافيهذهالح
                              ياةمتشبثينبالامناواحلامناوانكسارتنامالخاصة
                              Key: جحخربسصضدذزغقكطظعوؤإمنهأئةآىءشفيالت
Mutation gain:      -405.73   لاوقتللذكرياتولاالماضيالعالمليتذكرنابعدوسينسانابالرغممنذلكسنظلمعلقينهنافيهذهالح
                              ياةمتشبثينبالامناواحلامناوانكسارتناؤالخاصة
                              Key: جحخربسصضدذزغقكطظعوأإمنهؤئةآىءشفيالت
```

Figure 6.11: Part of a trace of the PPM cryptanalysis of an example Arabic Playfair cryptogram of 122 letters.

A second example, a sentence from ‘Al-Arab newspaper’ (https://alarab.co.uk): “كثرة الاهتمامات اليومية في البيت والعمل والضغوط التي ترافقها تؤدي كلها إلى التوتر والقلق الدائمين، كما تؤثر على جودة النوم ومدته، جميع هذه العوامل تشتت تركيز الدماغ وتتسبب في ضعف الذاكرة والتعرض مرارا إلى ظاهرة النسيان المفاجئ”, is also successfully solved by our model, (see Figure 6.12). Examples of the text at intermediate stages of the algorithm are shown in the figure, along with the compression codelength in bits. The correct solution with the best score is obtained from iteration 30.

```
Iteration: 0
2-Mutation gain:    -1019.53    خرجالاهتماماتالتكيتآسداأبدتحادهردحاسوبييالتدتانكخهالخيلمسهارسةالتحتآحالغد
                                صتدواصيتسهمالخصاأهدبعظيموزعذخذخوتذويتقذبالحقاملتشتتملوكفلدومابيتمتأنع
                                قششأهلالفاجكضالزقنشئاناويثفةتقوموزعكلاقالوشاخجو
...
Iteration: 6
                    -1009.66    كهمقالاهتمامايااليصديقشبالعبيكابحرآكالخحودالييبناأنهابصدلوبهاواةاليكتركالفبغزلمر
                                غديءومابصفشحبظشهدقالفنظنظرتثيديعسرحالطعامليجبتميروإسالمماحوتمتجارأتهخ
                                طيالضاقهوصالزخمإمنانا خشاةتخرقالفصلاذالنلاجةن
Iteration: 7
2-Mutation gain:    -966.42     قعفعالاهتماماتالتأرتنتياوريتقانهشيقالبنودالتيتراقبهالقدلحنهاشنةالتقتدقالمصاآثلرآمرتوهما
                                لقأهنوهفطاييوشوشاآتبصرتءمغتالزظاملتخاتميفاأسئالرمانوتمتعواءيسفهشالضاعثئوال
                                سحفجمراراحذيةيحفطاييجلاءالرشاأعر
...
Iteration: 27
Mutation gain:      -908.45     وممنالاهتماماتاليوميأطيانأيتوالبآلوالأسودالتيتراأئهاتؤديفلهابلنالتوترعالرجيجلحربميخفما
                                تؤفكلءكودباءفذبذبحتهسميثهذءالزواملتةيتمتدعمثالحماسوتمتغظآذاعنثلالقالعبوالث
                                مقمراراإنيشاهدباءفسياخالحياةبح
...
Iteration: 30
Mutation gain:      -569.68     كثرةالاهتماماتاليوميةفياالبيتوالعملوالضغوطالتيترافقهاتؤديكلهاإلىالتوتروالقلقالدائمينكما
                                تؤثرعلىجودةالنومومدتهجميعهذهالعواملتشتتتؤتركيزالدماغوتؤتسبؤبفيضعفالذاكرةوال
                                تعرضمراراإلىظاهرةالنسيانالمفاجئؤ
...
```

Figure 6.12: Example solutions produced by selected iteration for an Arabic cryptogram of 189 letters.

Part of the execution trace in the third example is shown in Figure 6.13. This example is a quote by ‘Ali ibn Abi Talib’: “إن أخوف ما أخاف عليكم طول الأمل ، واتباع الهوى : فأما طول الأمل فينسي الآخرة ، وأما اتباع الهوى فيصد عن الحق ..ألا وإن الدنيا قد تولت مدبرة ، والآخرة قد أقبلت مقبلة ، ولكل واحدة منهما بنون ، فكونوا من أبناء الآخرة ولا تكونوا من أبناء الدنيا ، فإن اليوم عمل ولا حساب ، والآخرة حساب ولا عمل”. The valid decrypt for this long cryptogram is obtained by the first iteration with the best codelength value `-607.69`.

```
Iteration: 0
Mutation gain:     -655.15     إناخوفماأخافعليكمطولالأملواتباعالهوىفأمارهلالأملفينسيالآخرةوأماماتباعالهوىفي
                               صدعنالحألاوإنالدنياقدتولتمدذغةوالآخرةقدأقبلتمقبلةولكلواحدةمنهمابنونفكونوا
                               منأبناءالآخرةولاتكونوامنأبناءالدنيافإناليومعملولاحسابوالآخرةحسابولاعم
Mutation gain:     -644.88     إناخوفماأخافعليكمطولالأملواتباعالهوىفأماطولالأملفينسيالآخرةوأماماتباعالهوىفي
                               صدعنالحقألاوإنالدنياقدتولتمدذزةوالآخرةقدأقبلتمقبلةولكلواحدةمنهمابنونفكونوا
                               منأبناءالآخرةولاتكونوامنأبناءالدنيافإناليومعملولاحسابوالآخرةحسابولاعمل
Mutation gain:     -607.69     إناخوفماأخافعليكمطولالأملواتباعالهوىفأماطولالأملفينسيالآخرةوأماؤاتباعالهوىفي
                               صدعنالحقألاوإنالدنياقدتولتمدبرةوالآخرةقدأقبلتمقبلةولكلواحدةمنهمابنونفكونوا
                               منأبناءالآخرةولاتكونوامنأبناءالدنيافإناليومعملولاحسابوالآخرةحسابولاعمل
```

Figure 6.13: Example solutions produced for an Arabic cryptogram of 221 letters.

The overall results of the main decryption phase ‘Phase I’ are presented in Table 6.21. This table exhibits the number of ciphertexts with different lengths that we have experimented with. The success rates achieved are also stated in this table. The results indicate that 51 ciphertexts out of 70 were effectively decrypted using our new method, and examples longer than 250

Table 6.21: Results when testing different Arabic Playfair ciphertexts with different lengths.

Cipher Length	**100-149**	**150-199**	**200-249**	**250-299**	**300-500**
No. of Ciphers	19	20	17	8	6
Success Rate (%)	53	65	82	100	100

letters were all successfully solved with no errors. However, short Arabic cryptograms of 110 letters or less were unable to be solved.

Concerning the second phase of our approach, which focuses on inserting spaces to the solved ciphertext that is outputted from Phase I, Figure 6.14 plots the segmentation results. Overall, these results indicate that most of the Arabic texts were efficiently segmented with no errors or with few errors. Figure 6.15 presents samples of segmented texts.

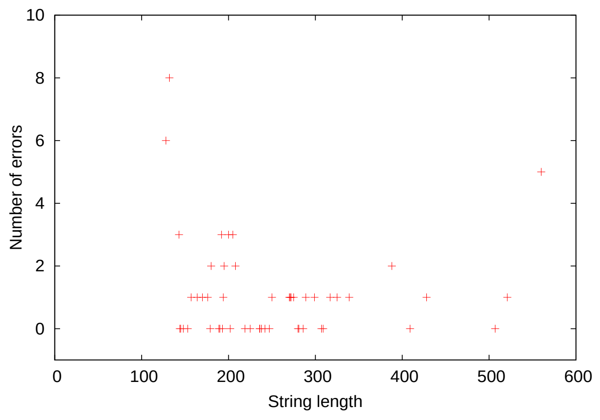


Figure 6.14: Segmentation errors produced as a result of the Phase II algorithm for the Arabic Playfair.

The high recall and precision rates and the low error rate produced by our segmentation method are listed in Table 6.22.

Table 6.22: Recall, precision and errors rates produced by our segmentation method for the Arabic decrypted texts produced from Phase I.

Recall (%)	**Precision (%)**	**Errors (%)**
95.23	95.40	4.77

The execution time needed to break the Arabic Playfair ciphertexts is almost double the time needed to crack the English texts. This is due to double the number of keys that were tested for each temperature used for Arabic and the preprocessing and postprocessing operations required when using the Arabic-specific encoding method. Table 6.23 lists the decryption time in seconds for Phase I of our method.

Table 6.23: Decryption times for different Arabic Playfair ciphertexts.

Ciphertext Length *(Letters)*	**122**	**189**	**221**	**365**	**500**
Time *(Sec)*	79	45	248	195	183

Ciphertext	ءأقكأوواهجأسزةءأأءلنصاأأءأطمهمهأعسودحجاأثكذإسانحناأحهخظهؤإاوبيامهاأقكعحمحنصتوحلحها أسؤوعفعرإساحءأؤإذلنذءأؤإذلنحيحخإذنلأءضأفى
Decrypted text	لا وقتل لذكريات ولا الماضي العالم لم يتذكرنا بعد وسينس انا بالرغم من ذلك سنظل معلقين هنا في هذه الحياة متشبثين بالامنا واحلامنا وانكسارتنا الخاصة
Ciphertext	اناإظصضحنضشخوضساإحءحضهجطنحثظظسسضوفكضبياحخغيحانحثظظسسضوفكضصفط صصفذؤإؤإظقخحإمكذإإدحننكاناإظصضحنضشخوضصأنبآأءضقناطظنحثظظسنغاأضشخوضسض فسضأحتظأحأطانضفسضحسعرظشأحانضغأطؤه
Decrypted text	لايهمني ان تكون ابيض اسود لايهمني ان تكون عربي اجنبي لايهمني ان تكون مسلم مسيحي يهودي بوذي شيعي الخ لايهمني ان تكون معارض او موالي ما يهمني فقط ان تكون انسان باعتبار الانسانية بدأت بالانقراض
Ciphertext	قجزئاءحلإونلذأأءتككإذوأسهاأعلذقاهملذءلفهسنوأأعخحشولحإةجإالملحامءسوأةززهاءكملطمخحء كإسملناإةجزأقسصمزءذماقهقهزإكركإبقحوأءتقنلوأفعإوبزبجخذمخلنفهإويذإهتغأصتشأءطنهجذةأءعي خغهدحخناأكءضنغءذمايجنحاأءوظسخؤه
Decrypted text	كثرة الاهتمامات اليومية في البيت والعمل والضغوط التي ترافقها تؤدي كلها إلى التوتر والقلق الدائمين كما تؤثر على جودة النوم ومدته جميع هذه العوامل تش تت تركيز الدماغ وت تسب ب في ضعف الذاكرة والتعرض مرارا إلى ظاهرة النسيان المفاجئ
Ciphertext	ؤإأءطهحءلطمهاكهإلحاإأءنحانحمقلحءحنزيلنياأءأأيعصاإهغزرمأهإزشتئهجبفزلحكءأءياأظقبشبجوهاإإحأ ادسحكءخحمأطخثإهغزرمإطأءةوأاربسخذأوقنطسلنحاأءسخإوطاأأءنحانساإغخهزأكماأحانحاقءسكض ياأخنمهنختصوقصتذظانمقوعغخهزؤوإطكتؤو
Decrypted text	من الحقائق المسلم بها أن الانسان هو كائن اجتماعي لا يعيش بمفرده بل تدفعه غريزته الى العيش مع غيره من أبناء جنسه لأنه عاجز بمفرده عن الوفاء بحاجاته واذا كان الاجتما ع الانساني ضروري للإنسان على الصعيد المادي فهو في نفس الوقت ضرورة معنوية
Ciphertext	دإخضةولنخضذضاأقجقنقماأءإلمقذأأحاأطموكةضتلنذقءأءخهمصتحاأسقءدزذةإلنلذأأحاأطموكةصت ظجإطاأءطهخءذلدإأءظإأسمثزةوأؤظرغذةأءءثزئمثعلاأهإوهعوءقململذنرؤوحملنإحمذصومذلذؤإإتحناأق ءدزذةءأيومذلذؤإإتحناأمخسإذضدإأءتكقإهمقمنذانتهاأءءثزئذاحاأقمطاأهم
Decrypted text	إن أخوف ما أخاف عليكم طول الأمل واتباع الهوى فأما طول الأمل فينسي الآخرة وأما ا تباع الهوى فيصد عن الحق ألا وإن الدنيا قد تولت مدبرة والآخرة قد أقبلت مقبلة ولكل واحدة منهما بنون فكونوا من أبناء الآخرة ولا تكونوا من أبناء الدنيا فإن اليوم عمل ولا حساب والآخرة حساب ولا عم ل

Figure 6.15: Examples of solved Arabic ciphertexts with spaces inserted after Phase II.

6.8 Comparison with English Experiments

The purpose of this section is to compare the results that were conducted in this chapter with the results of the English experiments that have been obtained in chapters 3, 4 and 5.

Based on Teahan (1998), the entropy for English using PPM compression was estimated as 1.812 bpc. This compares with 1.923 bpc that we have obtained for the Arabic language. These results were based on using a large corpus of classical and modern Arabic texts of 513 megabytes size compared to a much smaller English training text of size 35 megabytes. However, even though a larger training text would probably lead to a significant improvement, the result we obtained was not as good as the English experiment.

Based on this result, there is some indication that the Arabic language is potentially a more challenging language for cryptanalysis which is also supported by the experimental results in sections 6.5.2, 6.6.2 and 6.7.2. The reason for this is that Arabic is a richer morphological language compared to English. The results indicate that substantially more training data is required to get performance comparable to English and we have used this insight in order to design our Arabic cryptanalysis systems.

The Arabic language is one of the most used languages in the world, but unfortunately the techniques used to encrypt and decrypt information in the Arabic language are rare with few publications. In particular, we have not been able to find any cryptographic system, either previously used or currently, that have researched these three main ciphers in Arabic: simple substitution, transposition and Playfair.

According to Arabic substitution ciphers, an efficient combination between the character-based model and the word-based model (when the CSA-PPM variation was used) resulted in the best cryptanalysis performance. The results showed that 72% of the Arabic cryptograms were successfully solved without any errors and over 91% were decrypted with just three errors or less. This compares with 92% of the cryptograms correctly decrypted without any errors and 100% with just three errors or less for the automatic cryptanalysis of simple substitution ciphers for the English language as shown Chapter 3.

Referring to Arabic transposition ciphers, 97% of the cryptograms were decrypted without any errors using the different CSA-PPM models with a high level of performance in segmenting words. This compares with 100% achieved for the automatic cryptanalysis of transposition ciphers for the English language as seen in Chapter 4.

For the Arabic Playfair ciphers, 51 ciphertexts out of 70 were successfully decrypted and segmented; examples longer than 250 letters were all solved with no errors. This compares with 65 English ciphertexts out of 75 that were efficiently solved and 100% of ciphers of length greater than 120 that were decrypted, as shown Chapter 5.

6.9 Conclusions

In this chapter, Arabic simple substitution, transposition and Playfair ciphers are introduced and the automatic cryptanalysis of these ciphers are also investigated. An experimental analysis of an adapted approach to the automatic decryption of these Arabic ciphers based on the PPM compression scheme has been conducted. In addition, this has also been compared with the Gzip compression method for the first two cipher systems. The CSA-PPM compression variants showed better performance than Gzip in both decryption and segmentation processing. The CSA-PPMC models showed slightly worse results than the CSA-PPMD models. Again, the CSA-PPM without update exclusions method, which typically shows slightly worse performance at the compression task, performs better at decryption here. Various Arabic ciphertexts with different lengths have been successfully decrypted and effectively segmented.

The different corpora that we have built and experimented with are also described in this chapter. Arabic entropy is estimated in order to measure the effectiveness of compression models and to use in our cryptanalysis investigations. The codelength ratio has been calculated for the raw Arabic texts, the 'CSA-PPM' and the 'BA-PPM' methods. We have found how the use of the adaptive PPM compression models for the Arabic language is more effective than the use of standard PPM for the Arabic language in terms of compression rate.

Concerning Arabic substitution ciphers, different Arabic corpora types with different sizes as training texts were examined. The large Mixed corpus, which is a large combination of classical and modern Arabic texts, has proven to be an effective corpus to be used in the automatic cryptanalysis of Arabic ciphers. The results also showed that only when using the new CSA-PPM variation (CSA-PPM without update exclusions with the same specific codelength value assigned to all order -1 context predictions) results in successfully decrypted texts being produced. Character-based PPM compression models, bigraph-based models and word-based models were also explored in our experiments.

Referring to Arabic transposition ciphers and according to the main decryption phase, the CSA-PPM method with no update exclusions showed a better performance than the standard CSA-PPM models with 97% of the Arabic cryptograms were successfully decrypted with no errors comparing to 95%.

A variation of an order 5 CSA-PPMD model without update exclusions has been used in the cryptanalysis of the Arabic Playfair cipher. This variation has proven to be the most effective method for automatically recognising the valid decryption for Arabic Playfair ciphers.

It is worth noting that the data that was used to train the PPM models adapted for the Arabic language is significantly larger (more than 500 megabytes) than the much smaller training texts that were used for English (just 20 megabytes). This difference is significant especially because the larger model should give a better approximation to the language. However, even with the use of the large training model for Arabic, we were not able to achieve the same decryption results that were achieved for English.

In conclusion, experiments have confirmed that the compression-based cryptanalysis method works for another language, Arabic that is non-related to English, while the only previous experiments have been done for English. The results also highlight why we need to apply our new cryptanalysis method to other languages, and the Arabic language specifically, with indication that it is potentially a more challenging language for cryptanalysis and more difficult than for English.

Chapter 7

Conclusions

7.1 Introduction

This chapter discusses the work reported in this thesis and the experiments that have been performed. It also highlights the most important results, while reviewing the research questions and the study's initial aim and objectives. Suggestions for future research along with personal and professional recommendations are addressed at the end of this thesis.

7.2 Summary and Conclusions

Many electronic communications are compressed and encrypted separately before transmission. It is recognized that compression before transmission not only reduces transmission cost but also provides greater security. This study investigated the feasibility of using compression-based approaches in the field of cryptanalysis. Investigating the effectiveness of compression in tackling the plaintext recognition problem for cryptanalysis was the main concern of this study. Several new methods for the automatic cryptanalysis of classical ciphers using compression have been proposed. Considerable improvements in decipherment accuracy have been achieved. In general, automatic cryptanalysis of classical ciphers using compression is part of the wider field of the study of the history of cryptography and particularly of the study of historical ciphers. This provides a deeper understanding of the history of the development of both ciphers and cryptanalysis methods.

This research firstly reviewed classical cryptology. Classical systems were primarily discussed whereas modern ciphers were beyond the research scope. The historical development of ciphers and attacks were explained. Secondly, modern compression systems that tend to use adaptive models have been described. An extensive discussion of the prediction by partial matching (PPM) compression scheme and how the codelength metric is calculated have also been subsequently presented. We have surveyed different PPM models and have compared major PPM variants. This was followed by a review of how compression is usually used for cryptology. As experimenting with a language non-related to English was one of this study's main objectives, several fundamental characteristics of the Arabic language and its encoding methods have been described. In addition, the PPM compression scheme for the Arabic language has been explained.

In Chapter 3, the use of three compression schemes—PPM, Gzip and Bzip2—were explored to design an automatic compression-based cryptanalysis method against simple substitution ciphers. The basic idea of our approach depended on using these compression schemes as the basis for calculating the compression codelength metric which is an accurate way of measuring information in the text. The calculation of the codelength metric is described in detail in this chapter. Extensive investigations were performed to determine the most appropriate type of PPM scheme that could be applied to the problem of automatically breaking substitution ciphers. A new character-based PPM variant was proposed for the automatic decryption of this cipher. An efficient combination of this new variant and a beam-style search algorithm was introduced in this chapter. The experimental results showed how the newly devised PPM method significantly outperformed other schemes including the standard Gzip and Bzip2 compression schemes. Due to the nature of the Bzip2 scheme, some of the codelength calculations ended up being negative, making the use of this scheme unfeasible. The PPMD compression model tended to perform slightly better than the PPMC method. We also applied a word-based PPM variant which, when combined with the character-based method, led to further improvement in results. Experimental results showed that about 92% of the cryptograms were decrypted correctly without any errors and 100% with three errors or less.

Chapter 4 investigated the use of the PPM and Gzip compression schemes in the automatic cryptanalysis of transposition ciphers for the English language. The approach used in this investigation was similar to that used in Chapter 3. A compression-based approach was used as the basis for calculating the codelength value and then recognizing the valid decryption. As spaces are traditionally omitted from ciphertext, automatically achieving readability using compression methods was also proposed in this chapter. The Gzip compression scheme tended to perform worse than the PPM scheme on both decryption and segmentation processes. Ciphertexts of different sizes (ranging from 12 to 600 letters) were tested. A 100% decryption success rate was achieved by using the PPM compression models, in addition to the high-quality performance achieved in segmenting the decrypted texts. Again, slightly better results were obtained in the word segmentation step using PPMD models than for PPMC models.

Chapter 5 applied the PPM compression model to the problem of automatically decrypting Playfair ciphers. This cipher is generally considered to be more secure and cryptanalytically challenging than the previously examined ciphers. An efficient combination of text compression and simulated annealing was proposed for the automatic cryptanalysis of this cipher. The experimental results showed that the performance of the PPM-based cryptanalysis method was superior to previously published methods. The method was tried on various cryptograms of different lengths (starting from 60 letters) and a substantial majority of these ciphers were rapidly solved without any errors, with 100% of ciphers of lengths over 120 letters being solved. In efforts to achieve readability, the solutions were ranked, again using the PPM compression models. Almost all the decrypted examples were effectively segmented with a low average number of errors. Furthermore, we were able to break a Playfair cipher for a 6×6 grid using the newly devised method. To date, and in terms of its performance, this attack is state of the art.

In Chapter 6, we further investigated our English-based cryptanalysis work to examine if it was applicable to another language and to justify whether the Arabic language would provide a greater challenge for decryption. We also specifically chose this language as it has characteristics that differentiate it from other languages and as Arabic is non-related to English. This chapter introduced new compression-based methods for the effective automatic cryptanalysis of Arabic simple substitution, transposition and Playfair ciphers. A PPM model was successfully used as the basis for these methods. The use of the Gzip compression scheme was also investigated in this chapter with less efficient performance demonstrated by this method. To achieve readability, two more compression-based approaches for the insertion of spaces were evaluated with a high success rate achieved. To the best of the author's knowledge, this is the first work to demonstrate an effective automatic cryptanalysis for these three ciphers in Arabic. In addition, the entropy of the Arabic language was estimated in this chapter.

A comparison between the Arabic experimental results and the English results was also provided in Chapter 6. The results showed that 72% of the Arabic simple substitution cryptograms were successfully solved without any errors and over 91% were decrypted with three errors or less. Methods for English were found to be more likely to perform well than those for Arabic, with 92% of the cryptograms correctly decrypted without any errors and 100% with three errors or less. Regarding transposition ciphers, a success rate of 97% was achieved by using different Arabic PPM compression variants. This compares with the 100% success rate achieved in the automatic cryptanalysis of transposition ciphers for the English language. The results for Playfair ciphers showed that 73% of the Arabic ciphertexts were effectively decrypted and segmented; ciphers longer than 250 letters were all solved with no errors, while a success rate of 87% was achieved for the English Playfair ciphers with 100% success achieved for the decryption of examples longer than 120 letters. The size of the training texts used to prime the PPM models adapted for the Arabic language was more than 500 megabytes (MB) while the size used for English was only 20 MB. Even though the use of a larger model should give a better approximation to the language, our study's results indicated that we could not achieve the same decryption results in Arabic as we achieved for English. This suggests that Arabic language ciphers are more difficult to break.

7.3 Review of Research Questions

The research questions of this study, as listed in Section 1.2, have all been addressed. The research work illustrated the successful application of compression in the domain of automatic cryptanalysis of classical ciphers with challenging settings. Some cryptanalysis methods described in this thesis involved novel methods not previously described in the literature. The PPM compression scheme that has performed well in different language modelling tasks can also be successfully applied to the automatic decryption of these classical ciphers in both English and Arabic.

The specific research questions detailed in Section 1.2 have been addressed as follows:

- *Can compression models be used for effective cryptanalysis? Specifically, can we develop new effective methods for the automatic cryptanalysis of simple substitution, transposition and Playfair ciphers using compression?*

  Results in Chapters 3, 4 and 5 confirmed that the PPM compression scheme produced outstanding results for all three ciphers. These results presented strong evidence that compression models can be extremely effective for cryptanalysis. Chapters 3, 4 and 5 introduced new effective cryptanalysis methods and their variants based on compression. The results of simple substitution ciphers showed that approximately 92% of the cryptograms were correctly solved with no errors, with 100% being decrypted with just three errors or less. A 100% decryption success rate was achieved for transposition ciphers and 87% for Playfair ciphers. Moreover, using this efficient approach, successful decryption of an extended Playfair cipher for a 6×6 key matrix was achieved.

- *Does the prediction by partial matching (PPM) compression method perform better than other common compression methods for cryptanalysis?*

  Compared to other compression methods, namely, Gzip and Bzip2, the PPM scheme achieved much better results with high accuracy. In particular, the PPM method without update exclusions, which generally showed slightly worse performance at the compression task, was found to be the most effective variant at decryption. The Gzip method showed very poor performance at the decryption of simple substitution ciphers but good results were obtained for transposition ciphers. A third compressor, Bzip2, could not be used, as shown in Chapter 3, as, due to the nature of that scheme, some codelength calculations ended up being negative. However, none of the positive results for Bzip2 showed much success, with a high number of errors.

- *Can the newly devised methods be applicable to a language non-related to English (specifically Arabic) and how effective are these methods?*

  Experiments reported in Chapter 6 confirmed that the compression-based cryptanalysis approach worked well for Arabic, a language non-related to English, while previous experiments had only been conducted for English. Very promising results were obtained using the newly devised PPM-based cryptanalysis methods adapted specifically for Arabic language ciphers. For Arabic simple substitution ciphers, 72% of the cryptograms were successfully solved without any errors, and over 91% were decrypted with just three errors or less. For Arabic transposition ciphers, a success rate of 97% was achieved, with this result being 73% for Arabic Playfair ciphers.

- *Can compression models be used effectively to achieve readability of decrypted texts for cases when spaces have been omitted from ciphertexts?*

  Very effective word segmentation methods using PPM compression models have been used, with excellent results achieved as shown in Chapters 4, 5 and 6. The Gzip scheme was found to be ineffective in terms of segmentation performance for both English and Arabic, as presented in Chapters 4 and 6.

7.4 Review of Aim and Objectives

The aim and objectives of this study, as outlined in Section 1.3, have all been successfully achieved. The study has proposed new techniques to automatically break simple substitution, transposition and Playfair ciphers using compression, mainly PPM. These methods have proven to be very effective in terms of performance. With these new attacks, the current study has been able to decipher several challenging cryptograms, such as very short messages.

More specifically, the objectives, as presented in Section 1.3 that were used to investigate the study's research questions, have been achieved as follows:

- *Produce a literature review on the area of cryptology and compression with a specific focus on the relationship between them.*

This objective has been achieved in Chapter 2 in which the general concepts of cryptology and compression were presented. Different cipher systems and various techniques of text compression were reviewed and the use of compression in cryptology was addressed.

- *Develop new compression-based cryptanalysis methods for the automatic decryption of simple substitution, transposition and Playfair ciphers using PPM, and compare their effectiveness with alternative compression schemes such as Gzip and Bzip2.*

  This objective was achieved in Chapters 3, 4 and 5. To achieve this goal, new methods were developed for effective cryptanalysis of these three classical ciphers using compression. The performance of the PPM model was superior to the other standard compression schemes, Gzip and Bzip2.

- *Evaluate and validate the newly devised cryptanalysis methods using an extensive set of cryptograms, especially the more challenging cases, such as short ciphertexts, and investigate the effectiveness of these methods.*

  This objective was achieved in Chapters 3, 4 and 5. The wide range of case studies adopted in this research were used to evaluate and validate the newly devised methods. The new PPM-based methods have proven to be very effective in the cryptanalysis of these cases with state-of-the-art results produced. This study has included the decryption of classical ciphers within challenging settings, such as very short ciphertexts with no probable words. It has also included cryptograms published by the American Cryptogram Association and Royal New Zealand (NZ) Navy historical ciphertexts, as well as cryptograms published by geocache enthusiasts.

- *Construct a full cryptanalysis mechanism which also automatically adds spaces to decrypted texts, again using a compression-based ap-*

proach, to improve readability.

This objective was achieved in Chapters 4, 5 and 6. A full cryptanalysis mechanism was constructed consisting of two main parts: automatically decrypting ciphertexts and then automatically achieving readability. Compression models were used as a basis for these two parts. Readability was achieved through the development of new PPM-based methods that automatically segmented the decrypted outputs by reinserting spaces.

- *Develop and evaluate new compression-based decryption methods adapted for the automatic cryptanalysis of Arabic ciphers: simple substitution, transposition and Playfair ciphers in Arabic, and investigate whether Arabic, which is a language non-related to English, is more difficult for cryptanalysis purposes.*

  This objective was achieved in Chapter 6. New methods, especially adapted for the automatic cryptanalysis of these three Arabic ciphers using compression, have been successfully developed. In order to measure the effectiveness of PPM compression models and to facilitate the cryptanalysis investigation, the entropy of Arabic was estimated. These models were trained on a large Arabic corpus that was built specifically for this purpose. The results achieved showed a slightly worse performance for Arabic than for English, with an indication that Arabic is potentially a more challenging language for cryptanalysis than English.

7.5 Future Work

Based on this research, several questions have arisen which deserve further investigation. The issues can be summarized as follows:

- The use of larger training text should be explored. This should lead to further improvement in modelling which leads to improved attacks against other ciphers.

- Further extensions to our automatic cryptanalysis methods and an investigation into its applicability to other ciphers are required. Possibilities include extending the automatic attack to Vigenère, Beaufort, ADFGVX, bifid, homophonic and polyalphabetic substitution ciphers. On the other hand, it is not clear whether the compression-based cryptanalysis approach is applicable to modern ciphers and further investigation should be performed.

- The use of the PPM compression-based approach for the task of identifying the ciphertext language, and the problem of identifying the encryption algorithm which generated the ciphertext needs to be explored.

- Further investigation needs to be performed regarding other Arabic ciphers and the automatic decryption of these ciphers to ascertain whether this language really provides a greater challenge for cryptanalysis.

- The decipherment of some historical cases and challenging classical ciphers that have not yet been solved, such as German diplomatic and Navy messages, would be useful to investigate.

- It would be nice to explore other different search algorithms also based on the compression approach for the automatic decryption of the three main classical ciphers: simple substitution, transposition and Playfair ciphers.

References

Al-Kadit, I. A. (1992). Origins of cryptology: The Arab contributions. *Cryptologia*, 16(2):97–126.

Al-Kazaz, N. R., Irvine, S. A., and Teahan, W. J. (2016). An automatic cryptanalysis of transposition ciphers using compression. In *the 15th International Conference on Cryptology and Network Securit*, pages 36–52. Springer Int. Publishing. (https://link.springer.com/chapter/10.1007/978-3-319-48965-0_3), (accessed 11 December 2016).

Al-Kazaz, N. R., Irvine, S. A., and Teahan, W. J. (2018a). An automatic cryptanalysis of Playfair ciphers using compression. In *Proceedings of the International Conference on Historical Cryptology (HistoCrypt2018)*, number 149, pages 115–124. Linköping University Electronic Press. (http://www.ep.liu.se/ecp/article.asp?issue=149&article=021&volume=), (accessed 1 July 2018).

Al-Kazaz, N. R., Irvine, S. A., and Teahan, W. J. (2018b). An automatic cryptanalysis of simple substitution ciphers using compression. *Information Security Journal: A Global Perspective*, 27(1):57–75. Taylor & Francis. (https://www.tandfonline.com/doi/abs/10.1080/19393555.2018.1426799), (accessed 2 March 2018).

Alghamdi, M. A. and Teahan, W. J. (2017). A new thinning algorithm for Arabic script. *International Journal of Computer Science and Information Security*, 15(1):204–211.

Alhawiti, K. M. (2014). *Adaptive models of Arabic text.* PhD thesis, Bangor University.

Alkahtani, S. (2015). *Building and verifying parallel corpora between Arabic and English.* PhD thesis, Bangor University.

Alkahtani, S., Liu, W., and Teahan, W. J. (2015). A new hybrid metric for verifying parallel corpora of Arabic-English. In *the 5th International Conference on Computer Science, Engineering and Applications (CCSEA-2015)*, Dubai, UAE. AIRCC Publishing Corporation.

Alkhzaimi, H. A. (2016). *Cryptanalysis of selected block ciphers.* PhD thesis, Technical University of Denmark (DTU).

Almahdawi, A. and Teahan, W. J. (2017). Emotion recognition in text using PPM. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 149–155. Springer Int. Publishing.

Alqahtani, Y., Kuppuswamy, P., and Shah, S. (2013). New approach of Arabic encryption/decryption technique using Vigenere cipher on mod 39. *International Journal of Advanced Research in Information Technology and Engineering*, 2(12):1–9.

Alrabiah, M., Al-Salman, A., and Atwell, E. (2013). The design and construction of the 50 million words KSUCCA. In *Proceedings of the Second Workshop on Arabic Corpus Linguistics (WACL)*, pages 5–8. The University of Leeds.

Anderson, R. (1989). Finding vowels in simple-substitution ciphers by computer. In *Cryptology: Machines, History & Methods*, pages 215–225. Artech House, Inc.

Ball, W. R. (1960). *Mathematical Recreations and Essays.* New York, NY: Macmillan.

Bauer, C. P. (2016). *Secret history: the story of cryptology.* Chapman and Hall/CRC.

Bauer, C. P. (2017). Why history should matter to code breakers. *Scientific American, a Division of Nature America, Inc.*

Bell, T. C., Cleary, J. G., and Witten, I. H. (1990). *Text compression.* New Jersey: Prentice-Hall, Inc.

Bentley, J. L., Sleator, D. D., Tarjan, R. E., and Wei, V. K. (1986). A locally adaptive data compression scheme. *Communications of the ACM*, 29(4):320–330.

Bergen, H. A. and Hogan, J. M. (1993). A chosen plaintext attack on an adaptive arithmetic coding compression algorithm. *Computers & Security*, 12(2):157–167.

Bergmann, K. P., Scheidler, R., and Jacob, C. (2008). Cryptanalysis using genetic algorithms. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pages 1099–1100, New York, USA. ACM.

Biham, E. (1994). New types of cryptanalytic attacks using related keys. *Journal of Cryptology*, 7(4):229–246.

Bloom, C. (1998). Solving the problems of context modeling. *California Institute of Technology*, pages 1–11.

Bose, R. and Pathak, S. (2006). A novel compression and encryption scheme using variable model arithmetic coding and coupled chaotic system. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(4):848–857.

Broemeling, L. D. (2011). An account of early statistical inference in Arab cryptolog. *The American Statistician*, 65(4):255–257.

Budiansky, S. (2000). *Battle of wits: the complete story of codebreaking in World War II.* New York, NY: Simon and Schuster.

Burke, C. B. (2002). *It Wasn't All Magic: The Early Struggle to Automate Cryptanalysis, 1930s-1960s.* Center for Cryptologic History, National Security Agency.

Burrows, M. and Wheeler, D. (1994). A block-sorting lossless data compression algorithm. Technical report, Digital Equipment Corporation, Palo Alto, California.

Bzip2 (2016). The bzip2 home page. (http://www.bzip.org), (accessed 15 June 2016).

Chang, Z. (2008). A PPM-based evaluation method for Chinese-English parallel corpora in machine translation. Master's thesis, Bangor University.

Chen, J. and Rosenthal, J. S. (2012). Decrypting classical cipher text using Markov Chain Monte Carlo. *Statistics and Computing*, 22(2):397–413.

Chen, J., Zhou, J., and Wong, K.-W. (2011). A modified chaos-based joint compression and encryption scheme. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 58(2):110–114.

Clark, A. (1994). Modern optimisation algorithms for cryptanalysis. In *Proceedings of the Second Australian and New Zealand Conference on Intelligent Information Systems*, pages 258–262. IEEE.

Clark, A. (1998). *Optimisation heuristics for cryptology.* PhD thesis, Queensland University of Technology.

Cleary, J. G., Irvine, S. A., and Rinsma-Melchert, I. (1995a). On the insecurity of arithmetic coding. *Computers and Security*, 14(2):167–180.

Cleary, J. G., Teahan, W. J., and Witten, I. H. (1995b). Unbounded length contexts for PPM. In *Data Compression Conference (DCC'95), Proceedings*, pages 52–61. IEEE Computer Society Press.

Cleary, J. G. and Witten, I. H. (1984a). A comparison of enumerative and adaptive codes. *IEEE Transactions on Information Theory*, 30(2):306–315.

Cleary, J. G. and Witten, I. H. (1984b). Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402.

Comrie, B., Fabri, R., Hume, E., Mifsud, M., Stolz, T., and Vanhove, M. (2009). *Introducing Maltese Linguistics: Selected papers from the 1st International Conference on Maltese Linguistics.* Studies in Language Companion Series. Amsterdam: John Benjamins Publishing Company.

Copeland, B. J. (2010). *Colossus: The secrets of Bletchley Park's code-breaking computers.* Oxford University Press.

Corlett, E. and Penn, G. (2010). An exact A* method for deciphering letter-substitution ciphers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1040–1047. Association for Computational Linguistics.

Cowan, M. J. (2008). Breaking short Playfair ciphers with the simulated annealing algorithm. *Cryptologia*, 32(1):71–83.

Deavours, C. A. and Kruh, L. (1985). *Machine cryptography and modern cryptanalysis.* Artech House, Inc.

Delman, B. (2004). Genetic algorithms in cryptography. Master's thesis, Department of Computer Engineering, Rochester Institute of Technology.

Denning, D. E. (1982). *Cryptography and data security.* Boston: Addison-Wesley Longman Publishing Company, Inc.

Devi, A. and Mani, K. (2018). Enhancing security in RSA cryptosystem using Burrows-Wheeler transformation and run length encoding. *Inter-*

national Journal of Scientific Research in Computer Science, Engineering and Information Technology, 3(1):687–696.

Dimovski, A. and Gligoroski, D. (2003). Attacks on the transposition ciphers using optimization heuristics. In *Proceedings of the International Scientific Conference on Information, Communication & Energy Systems & Technologies (ICEST)*, pages 1–4, Sofia, Bulgaria.

Duan, L., Liao, X., and Xiang, T. (2011). A secure arithmetic coding based on Markov model. *Communications in Nonlinear Science and Numerical Simulation*, 16(6):2554–2562.

Dukes, K. and Habash, N. (2010). Morphological annotation of Quranic Arabic. In *Language Resources and Evaluation Conference (LREC). Valletta, Malta.*

Eskicioglu, A. and Litwin, L. (2001). Cryptography. *IEEE Potentials*, 20(1):36–38.

Esslinger, B. (2008). Cryptool 2.0. (https://www.cryptool.org/en/cryptool2), (accessed 1 September 2018).

Forsyth, W. S. and Safavi-Naini, R. (1993). Automated cryptanalysis of substitution ciphers. *Cryptologia*, 17(4):407–418.

Francis, W. and Kucera, H. (1982). *Frequency analysis of English usage.* Boston: Houghton Mifflin Company.

Friedman, W. F. (1976). *Elements of Cryptanalysis*, volume 3. Laguna Hills, CA: Aegean Park Press.

Friedman, W. F. and Lambros, C. D. (1985). *Military Cryptanalytics, Part I.* Laguna Hills, CA: Aegean Park Press.

Gaines, H. F. (1956). *Cryptanalysis: A study of ciphers and their solution.* New York, NY: Dover Publications.

Garg, P. and Sherry, A. (2005). Genetic algorithm and tabu search attack on the mono-alphabetic substitution cipher. *Paradigm*, 9(1):106–109.

Gary, S. F. and Fennig, C. D. (2018). *Ethnologue: Languages of the world (21st edition)*. Texas: SIL International.

Gavaskar, S., Ramaraj, E., and Surendiran, R. (2012). A compressed anti IP spoofing mechanism using cryptography. *International Journal of Computer Science and Network Security*, 12(11):137–140.

Giddy, J. and Safavi-Naini, R. (1994). Automated cryptanalysis of transposition ciphers. *The Computer Journal*, 37(5):429–436.

Grundlingh, W. and Van Vuuren, J. H. (2003). Using genetic algorithms to break a simple cryptographic cipher. Retrieved 31 March 2003 from (http://dip.sun.ac.za/˜vuuren/abstracts/abstrgenetic.htm), submitted 2002.

Gzip (2012). The gzip home page. (http://www.gzip.org), (accessed 1 March 2016).

Habash, N., Soudi, A., and Buckwalter, T. (2007). On Arabic transliteration. In *Arabic Computational Morphology*, pages 15–22. Springer, Dordrecht.

Habeeb, R. S. (2016). Arabic text cryptanalysis using genetic algorithm. *Iraqi Journal for Electrical and Electronic Engineering*, 12(2):161–166.

Hammood, D. A. (2013). Breaking a Playfair cipher using memetic algorithm. *Journal of Engineering and Sustainable Development*, 17(5):172–183.

Hans, S., Johari, R., and Gautam, V. (2014). An extended Playfair cipher using rotation and random swap patterns. In *International Conference on Computer and Communication Technology (ICCCT)*, pages 157–160. IEEE.

Hart, G. W. (1994). To decode short cryptograms. *Communications of the ACM*, 37(9):102–108.

Hilton, R. (2012). *Automated cryptanalysis of monoalphabetic substitution ciphers using stochastic optimization algorithms.* PhD thesis, Department of Computer Science and Engineering, University of Colorado, Denver.

Horapollo (1950). *Hieroglyphica.* Translated by George Boas.

Horspool, R. and Cormack, G. (1986). Dynamic Markov modelling—a prediction technique. In *Proceedings of the International Conference on System Sciences.* Honolulu, HA, IEEE Computer Society Press.

Horspool, R. N. and Cormack, G. V. (1992). Constructing word-based text compression algorithms. In *Data Compression Conference*, pages 62–71, Snowbird, Utah. IEEE Computer Society Press.

Howard, P. G. (1993). *The design and analysis of efficient lossless data compression systems.* PhD thesis, Brown University, Providence, Rhode Island.

Howard, P. G. and Vitter, J. S. (1992). Practical implementations of arithmetic coding. In *Image and Text Compression*, pages 85–112. Springer.

Irvine, S. A. (1997). *Compression and Cryptology.* PhD thesis, University of Waikato, New Zealand.

Irvine, S. A., Cleary, J. G., and Rinsma-Melchert, I. (1995). The subset sum problem and arithmetic coding. *Department of Computer Science, University of Waikato.* Hamilton, New Zealand. (Working paper 95/7).

Ishibashi, H. and Tanaka, K. (2001). Data encryption scheme with extended arithmetic coding. In *Mathematics of Data/Image Coding, Compression, and Encryption IV, with Applications*, volume 4475, pages 222–234. International Society for Optics and Photonics.

Jain, A., Grover, J., and Jain, T. (2018). Cuckoo search strategies for solving combinatorial problems (solving substitution cipher: An investigation). In *Proceedings of the 1st International Conference on Smart System, Innovations and Computing*, pages 585–594. Springer.

Jakobsen, T. (1995). A fast method for cryptanalysis of substitution ciphers. *Cryptologia*, 19(3):265–274.

Jassim, M. K. (2017). Improved PSO algorithm to attack transposition cipher. *Engineering and Technology Journal*, 35(2):144–149.

Jasuja, B. and Pandya, A. (2015). Crypto-compression system: an integrated approach using stream cipher cryptography and entropy encoding. *International Journal of Computer Applications*, 116(21):34–41.

Kahn, D. (1967). *The Codebreakers: The Story of Secret Writing*. New York, NY: Macmillan.

Kahn, D. (1973). *The codebreakers*. London: Weidenfeld and Nicholson.

Kambhatla, N., Bigvand, A. M., and Sarkar, A. (2018). Decipherment of substitution ciphers with neural language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 869–874. Association for Computational Linguistics.

Katti, R. S., Srinivasan, S. K., and Vosoughi, A. (2011). On the security of randomized arithmetic codes against ciphertext-only attacks. *IEEE Transactions on Information Forensics and Security*, 6(1):19–27.

Katzner, K. (2002). *The languages of the world*. London: Routledge.

Kim, H., Wen, J., and Villasenor, J. D. (2007). Secure arithmetic coding. *IEEE Transactions on Signal Processing*, 55(5):2263–2272.

King, J. C. and Bahler, D. R. (1992). An implementation of probabilistic relaxation in the cryptanalysis of simple substitution ciphers. *Cryptologia*, 16(3):215–225.

Klima, R. E. and Sigmon, N. P. (2012). *Cryptology: classical and modern with maplets*. Florida: CRC Press.

Knight, K., Nair, A., Rathod, N., and Yamada, K. (2006). Unsupervised analysis for decipherment problems. In *Proceedings of the Computational Linguistics/Association for Computational Linguistics COLING/ACL on Main conference poster sessions*, pages 499–506. Association for Computational Linguistics.

Kodabagi, M., Jerabandi, M., and Gadagin, N. (2015). Multilevel security and compression of text data using bit stuffing and Huffman coding. In *International Conference on Applied and Theoretical Computing and Communication Technology (ICATCCT), 2015*, pages 800–804. IEEE.

Lasry, G. (2018). *A methodology for the cryptanalysis of classical ciphers with search metaheuristics*. Kassel University Press GmbH.

Lasry, G., Kopal, N., and Wacker, A. (2016). Cryptanalysis of columnar transposition cipher with long keys. *Cryptologia*, 40(4):374–398.

Lee, D. (2002). Substitution deciphering based on HMMs with applications to compressed document processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1661–1666.

Lee, K., Teh, C., and Tan, Y. (2006). Decrypting English text using enhanced frequency analysis. In *National Seminar on Science, Technology and Social Sciences*, pages 1–7.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710.

Lim, J., Boyd, C., and Dawson, E. (1997). Cryptanalysis of adaptive arithmetic coding encryption schemes. In *Australasian Conference on Information Security and Privacy*, pages 216–227. Springer.

Liu, W., Chang, Z., and Teahan, W. J. (2014). Experiments with a PPM compression-based method for English-Chinese bilingual sentence alignment. In *International Conference on Statistical Language and Speech Processing*, pages 70–81. Springer.

Lucks, M. (1990). A constraint satisfaction algorithm for the automated decryption of simple substitution ciphers. In *Advances in Cryptology (CRYPTO'88)*, pages 132–144. Springer.

Luthra, J. and Pal, S. K. (2011). A hybrid firefly algorithm using genetic operators for the cryptanalysis of a monoalphabetic substitution cipher. In *World Congress on Information and Communication Technologies (WICT)*, pages 202–206. IEEE.

Lyons, J. (2012). Cryptanalysis of the Playfair cipher. (http://practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-playfair/), (accessed 11 October 2016).

Matthews, R. (1993). The use of genetic algorithms in cryptanalysis. *Cryptologia*, 17(2):187–201.

Mauborgne, J. O. (1914). *An advanced problem in cryptography and its solution.* Fort Leavenworth, KS: Leavenworth Press.

Moffat, A. (1989). Word-based text compression. *Software: Practice and Experience*, 19(2):185–198.

Moffat, A. (1990). Implementing the PPM data compression scheme. *IEEE Transactions on Communications*, 38(11):1917–1921.

Mrayati, M., Alam, Y., and al-Tayyan, H. (1987). Ilm at-Ta'miyah wa Istikhraj al-Mu'amma ind al-Arab (Origins of Arab cryptography and cryptanalysis). Volume one. *Analysis and Editing of Three Arabic Manuscripts: Al-Kindi, Ibn-Adlan, Ibn-Al-Durahim.* Arab Academy of Damascus Publications, Damascus.

Muanalifah, A. (2017). Construction of key echange protocol over max-plus algebra to encrypt and decrypt Arabic documents. *Journal Of Natural Sciences and Mathematics Research*, 1(2):51–54.

Mudgal, P. K., Purohit, R., Sharma, R., and Jangir, M. K. (2017). Application of genetic algorithm in cryptanalysis of mono-alphabetic substitution cipher. In *International Conference on Computing, Communication and Automation (ICCCA)*, pages 400–405. IEEE.

Murali, P. and Senthilkumar, G. (2009). Modified version of Playfair cipher using linear feedback shift register. In *International Conference on Information Management and Engineering (ICIME'09)*, pages 488–490. IEEE.

Negara, G. (2012). An evolutionary approach for the Playfair cipher cryptanalysis. In *Proceedings of the International Conference on Security and Management (SAM), USA*, pages 8–14. Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Ng, T., Nguyen, K., Zbib, R., and Nguyen, L. (2009). Improved morphological decomposition for Arabic broadcast news transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4309–4312.

Nuhn, M., Schamper, J., and Ney, H. (2013). Beam search for solving substitution ciphers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1: Long Papers, pages 1568–1576. Association for Computational Linguistics.

Olson, E. (2007). Robust dictionary attack of short simple substitution ciphers. *Cryptologia*, 31(4):332–342.

Peleg, S. and Rosenfeld, A. (1979). Breaking substitution ciphers using a relaxation algorithm. *Communications of the ACM*, 22(11):598–605.

Ravi, S. and Knight, K. (2008). Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*, pages 812–819. Association for Computational Linguistics.

Ravi, S. and Knight, K. (2009). Attacking letter substitution ciphers with integer programming. *Cryptologia*, 33(4):321–334.

Ravindra, B. K., Kumar, S. U., Babu, A. V., Aditya, I. S., and Komuraiah, P. (2011). An extension to traditional Playfair cryptographic method. *International Journal of Computer Applications*, 17(5):34–36.

Rihan, S. D. and Osma, S. E. F. (2016). Arabic cryptography technique using neural network and genetic algorithm. *International Research Journal of Computer Science*, 3:35–42.

Rissanen, J. and Langdon, G. (1981). Universal modeling and coding. *IEEE Transactions on Information Theory*, 27(1):12–23.

Russell, M. D., Clark, J. A., and Stepney, S. (2003). Making the most of two heuristics: Breaking transposition ciphers with ants. In *the Congress on Evolutionary Computation (CEC'03)*, volume 4, pages 2653–2658.

Sacco, L. (1996). *Manual of Cryptography (Cryptographic Series)*. Laguna Hills, CA: Aegean Park Press.

Sandoval, M. and Feregrino-Uribe, C. (2005). A hardware architecture for elliptic curve cryptography and lossless data compression. In *Proceedings of the 15th International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 113–118. IEEE.

Sangwan, N. (2012). Text encryption with Huffman compression. *International Journal of Computer Applications*, 54(6):29–32.

Scharwächter, E. and Vogel, S. (2015). Solving substitution ciphers for OCR with a semi-supervised hidden Markov model. In *the 13th International*

Conference on Document Analysis and Recognition (ICDAR), pages 11–15. IEEE.

Schatz, B. R. (1977). Automated analysis of cryptograms. *Cryptologia*, 1(2):116–142.

Schneier, B. (1996). *Applied cryptography (2nd edition).* New York, NY: John Wiley and Sons, Inc.

Seberry, J. and Pieprzyk, J. (1989). *Cryptography: an introduction to computer security.* New Jersey: Prentice-Hall, Inc.

Shaban, S. A. and Najimaldin, B. (2017). A new algorithm for encrypting Arabic text using the mathematical equation. *Diyala Journal of Engineering Science*, 10(1):21–30.

Shannon, C. E. (1949). Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715.

Shannon, C. E. (1951). Prediction and entropy of printed English. *Bell System Technical Journal*, 30(1):50–64.

Sharma, R. and Bollavarapu, S. (2015). Data security using compression and cryptography techniques. *International Journal of Computer Applications*, 117(14):15–18.

Shkarin, D. A. (2001). Improving the efficiency of the PPM algorithm. *Problems of Information Transmission*, 37(3):226–235.

Singh, S. (2000). *The code book: the science of secrecy from ancient Egypt to quantum cryptography.* Anchor.

Sinkov, A. (1966). Elementary cryptanalysis: A mathematical approach. *Mathematical Association of America.* Random House, New York.

Smith, L. D. (1955). *Cryptography: The science of secret writing.* Massachusetts: Courier Corporation.

Spillman, R., Janssen, M., Nelson, B., and Kepner, M. (1993). Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers. *Cryptologia*, 17(1):31–44.

Srivastava, S. S. and Gupta, N. (2011). Security aspects of the extended Playfair cipher. In *International Conference on Communication Systems and Network Technologies (CSNT)*, pages 144–147. IEEE.

Stamp, M. and Low, R. M. (2007). *Applied cryptanalysis: breaking ciphers in the real world*. New Jersey: John Wiley and Sons.

Stumpel, J. (2007). Fast Playfair programs. (www.jw-stumpel.nl/playfair.html), (accessed 13 December 2017).

Sun, H. M., Wang, K. H., and Ting, W. C. (2009). On the security of the secure arithmetic code. *IEEE Transactions on Information Forensics and Security*, 4(4):781–789.

Teahan, W. J. (1998). *Modelling English text*. PhD thesis, University of Waikato, New Zealand.

Teahan, W. J. (2018). NLP using compression-based language models: Past, present and future [PowerPoint presentation]. (https://bangoroffice365-my.sharepoint.com/:p:/g/personal/eesa0e_bangor_ac_uk/EXL18rlYT8ZIr8cGySbW9WoBcCof4R8PQ4GeVg4pUENQGA?e=YAFT9o), (accessed 18 July 2018).

Teahan, W. J. and Alhawiti, K. (2013). Design compilation and preliminary statistics of compression corpus of written Arabic. Technical report, Bangor University.

Teahan, W. J. and Cleary, J. G. (1996). The entropy of English using PPM-based models. In *Data Compression Conference (DCC'96), Proceedings*, pages 53–62. IEEE.

Teahan, W. J. and Cleary, J. G. (1997). Models of English text. In *Data Compression Conference (DCC'97), Proceedings*, pages 12–21. IEEE.

Teahan, W. J. and Harper, D. J. (2001). Combining PPM models using a text mining approach. In *Data Compression Conference (DCC), Proceedings*, pages 153–162. IEEE.

Teahan, W. J., Wu, P., and Liu, W. (2014). Adaptive compression-based models of Chinese text. In *International Conference on Audio, Language and Image Processing (ICALIP)*, pages 874–881. IEEE.

Toemeh, R. and Arumugam, S. (2007). Breaking transposition cipher with genetic algorithm. *Elektronika ir Elektrotechnika*, 79(7):75–78.

Uddin, M. F. and Youssef, A. M. (2006). Cryptanalysis of simple substitution ciphers using particle swarm optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 677–680. IEEE.

Van Tilborg, H. C. (2012). *An introduction to cryptology*, volume 52. Springer Science & Business Media.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Vobbilisetty, R., Di Troia, F., Low, R. M., Visaggio, C. A., and Stamp, M. (2017). Classic cryptanalysis using hidden Markov models. *Cryptologia*, 41(1):1–28.

W3Techs (2013). Historical trends in the usage of character encodings for websites. Online: (https://w3techs.com/technologies/history_overview/character_encoding), (accessed 30 September 2017).

Wang, C. E. (2006). Cryptography in data compression. *Code-Breakers Journal, Security and Anti-Security-Attack and Defense*, 2(3):15–21.

Weekley, E. (2012). *An etymological dictionary of modern English*, volume 2. Massachusetts: Courier Corporation.

Wen, J., Kim, H., and Villasenor, J. D. (2006). Binary arithmetic coding with key-based interval splitting. *IEEE Signal Processing Letters*, 13(2):69–72.

Wiener, M. J. (1993). Efficient DES key search. In *Advances in Cryptology: CRYPTO'93, Lecture Notes in Computer Science*, volume 733. Berlin. Springer-Verlag.

Wikipedia (n.d.). CrypTool. (https://en.wikipedia.org/wiki/CrypTool), (accessed 29 June 2018).

Williams, E. (1959). *An invitation to cryptograms.* New York, NY: Simon and Schuster.

Wilson, W. J. (1994). Chinks in the armor of public key cryptosystems. Technical report 94/3, University of Waikato.

Witten, I. H. (1982). *Principles of computer speech.* London: Academic Press.

Witten, I. H. and Bell, T. C. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

Witten, I. H. and Cleary, J. G. (1988). On the privacy afforded by adaptive text compression. *Computers & Security*, 7(4):397–408.

Witten, I. H., Neal, R. M., and Cleary, J. G. (1987). Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540.

Wong, K. W., Lin, Q., and Chen, J. (2010). Simultaneous arithmetic coding and encryption using chaotic maps. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(2):146–150.

Wu, P. and Teahan, W. J. (2005). Modelling Chinese for text compression. In *Data Compression Conference (DCC), Proceedings*, pages 488–488. IEEE.

Wulandari, G. S., Rismawan, W., and Saadah, S. (2015). Differential evolution for the cryptanalysis of transposition cipher. In *the 3rd International Conference on Information and Communication Technology (ICoICT)*, pages 45–48. IEEE.

Xiang, T., Sun, J., and Fu, X. (2016). On the security of binary arithmetic coding based on interval shrinking. *Multimedia Tools and Applications*, 75(8):4245–4258.

Yilei, W., Xiaotao, C., Golander, A., and Franke, H. (2015). Internet-oriented optimization schemes for joint compression and encryption. *China Communications*, 12(10):158–168.

Zeki, A. M. (2005). The segmentation problem in Arabic character recognition the state of the art. In *the 1st International Conference on Information and Communication Technologies*, pages 11–26. IEEE.

Zhou, J., Au, O. C., Wong, P. H., and Fan, X. (2008). Cryptanalysis of secure arithmetic coding. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1769–1772. IEEE.

Ziv, J. and Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343.

Appendices

Appendix I: Samples of substitution cryptograms and their valid decrypted texts.

#	Cryptograms	Decrypted texts (produced by the software)	Source
1	vukjksrjeojvukjneiwr	the end of the world	Skeeter Davis (song)
2	fsdigzgdubisqmgfbmlgobmgisqzdtmtz	what s another word for thesaurus	Steven Wright
3	nmiyuljrtujqxaljcdjmoojylumrjbpinq	danger the spur of all great minds	George Chapman
4	zwkkebvaiivebknxbecpecnkebegntefacx	wall street slips of oil s big drop	New York Times
5	qdbyjjb dlvo juyzxdavkdz qkbfd e kfvz	a little something for nearly everyone	New York Times
6	mfzblzfxzofmzmfzblzmvpmzrdzmvlzecldmrfo	to be of not to be that is the question	William Shakespeare
7	zll jfuzqjrfrdbxorfbkfrjkbsifuzkzpjujko	apple makes shifts in senior management	New York Times
8	eyeixjayrxajdmzayakimxacytalwajlyrlahxwhpx	babies are such a nice way to start people	Don Herold
9	smzpdsilmnnhdyulgmsqjmhelcgpjmzulmccdhql pslnmhpi	nations approve landmark climate accord in paris	New York Times
10	zpvlnvedmoasqd olydlodjlnkdvedluydyadvcdnjhyqdloqnk	britain would vote to stay in the eu if asked today	The Independent
11	asmmgchejkt nsshmcmahejktzsnkhgsjcm hankxnkxhgzghazkxa	sleepy mandrill eyes mandolin player singing pop songs	Michael Lucks
12	rbu ih fdk fivk sbt cyy ebbw vkr fb gbvk fb fdk ciw bs fdkit pctfl	now is the time for all good men to come to the aid of their party	Charles E. Weller
13	rieitislkibadipdldqbehqhakdlamdabkdldrbamikib adldebclzhdlamdabkdldnl ybg	civilization is a movement and not a condition a voyage and not a harbour	Arnold Toynbee
14	iheunrkpkgreugeunerlkejzrzpketk vzgkeuevieomunoermegcknyerlkepkgremjeihed ujkerlkpk	my interest is in the future because i am going to spend the rest of my life there	Charles Kettering
15	epuluovluoeyxosbkkbxqobqeulu ebqtoiuxikuobqoquyofxlrovqdoxqkfo uhuqefoubtpeobqokx ovqtuku	there are two million interesting people in new york and only seventy eight in los angeles	Neil Simon
16	srnevwxnowernwvzniwpzmwdnnwvzevwdalrlv gekwldwdvrmosnrwvzeowxevnrlekwumrqnwvz evwvzmgszvdwrgknwvznwpmrkb	great men are they who see that spiritual is stronger than material force that thoughts rule the world	Ralph Waldo Emerson
17	sqhguuxqxgcqgsnnunuxk uvhqzsxgqtvaacxhq klcqueqnakxspv qtcwkvhcqseqsqvhcqacklcpqupchqgshqgslcqshq hvfcqxuqeakxxcpqc	i shoot the hippopotamous with bullets made of platinum because if i use leaden ones his hide is sure to flatten em	George Hart
18	yosnmgmcsnysccvnkvnyfncfxsnfkwndsgbomfw vnhdznhcvfnyfncfxsnfkwnsdslgsvnuwfmhmcqn msjhkvsnyosqnhwsnbsdswhccqnyosnvhlsnusfuc s	the bible tells us to love our neighbors and also to love our enemies probably because they are generally the same people	Gilbert Keith Chesterton
19	poltjdqmujzsumujmjs jblmtbjdljuqlcjrlpjqlcjsjzptaqjlijzhmbqdjrlptbjiln eujfmfjimtfjsjaqs omltjsj stjcmdqjzlrujstfjbmhnujlijqmujlct	upon this basis i am going to show you how a bunch of bright young folks did find a champion a man with boys and girls of his own	Ernest Vincent Wright
20	gjbuypbigbwgufsegplzkglseucbufieuvihbugsusk pudskeywgbpwuleuspobpugsu pb ipbugjbvutspugjbulebalgizrbulwugsulvzkbuleug jbvuiuybeklebuwbewbustuioabegkpbuclgjskgug jiguwbewbusebuvidubiwlrduobw ilpuieoucbuebboupigjbpuobw bpigbrdugjbu iglbefbuieoufskpiybustuvbeuieoucsvbeucjsupbtk wbugsuobw ilp	the greatest contribution we can make to our youngsters in order to prepare them for the inevitable is to imbue in them a genuine sense of adventure without that sense one may easily despair and we need rather desperately the patience and courage of men and women who refuse to despair	Edward Eddy

Appendix II: Samples of transposition cryptograms and their valid decrypted texts.

#	Cryptograms	Decrypted texts (produced by the software)	Source
1	lbbiobgsagni	bilbo baggins	Phrase
2	ogcyrptremahoydp	cryptography demo	Phrase
3	ousmhcosnusdlotiltoetuocem	so much sound so little outcome	The Guardian
4	iusictmsehomtsxeepsnvioeafllosisen	music is the most expensive of all noises	Josef Hofmann
5	awthitshucnoytnrdesemiosfrresepeceherwtoihsl ntietnog	what this country needs is more free speech worth listening to	Hansell Duckett
6	dkitelirhaeeofmhtsuterufeteetbhartnthihtseyootf rpahtse	i like the dreams of the future better than the history of the past	Thomas Jefferson
7	eillaficomnorswesteeidalnnoetdwdeyhtoodnluht acerlancocxuonis	if all economists were laid end to end they would not reach a conclusion	George Bernard Shaw
8	tehhnwisydaednaaacuhtgoiwitothedlubuhnietpt uoanmosnmsirhweee	when they said canada i thought it would be up in the mountains somewhere	Marilyn Monroe
9	oluiwdhtaretebrhnamewobohutthghorbeonylkb gdirenahttamehnsohwotidl	i would rather be the man who bought the brooklyn bridge than the man who sold it	Will Rogers
10	itrreetnmmueebswotrenfudemlanicuyanokcsinu ruystocaohfmnoolyrolsngo	retirement must be wonderful i mean you can suck in your stomach for only so long	Burt Reynolds
11	tttamaihhwergeaeuoalfyifenuotetdtlwealthlkisin reesnebedvttlleefa	im at that age where if you flattened out all the wrinkles i d be seven feet tall	Robert Orben
12	tcaiotmmheetiisahgncwthiekaweesahtkwoadoo otondegdamnnocaunioarnh	a committee is a thing which takes a week to do what one good man can do in an hour	Elbert Hubbard
13	atngsoeitaiawmsnhnohsitktfhteiahhbanetdenpbr onoepulwdoelhoaedwvnehrdxwey	a negotist is a man who thinks that if he hadn t been born people would have wondered why	Dan Post
14	ndilrceheneuparreactbdlireunvyeotwowhkansnc osiineynctthegoegiynrytcacthotnineoxu	children are unpredictable you never know what in consistency they re going to catch you in next	Franklin P. Jones
15	infaremnodfintweeteotohdrbtcoaaiuotrnignigsa hnieeotrhdratcogiaevhmlsunnieretda	a friend of mine went to the doctor about a ringing in his ear the doctor gave him an unlisted ear	Charlie Callas
16	ouacyneaadlhrsteooatrewbtbfeuoeyuorpshihum njsuittoapsnthnidkowwahehosrtemellss	you can lead a horse to water but before you push him in just stop and think how a wet horse smells	George Gobel
17	eystnidinsomtateorfcahncietiasmatterfochiocetii sontahtintgobweaietdfroitsiatihngotbedahiveec	destiny is no matter of chance it is a matter of choice it is not a thing to be waited for it is a thing to be achieved	William Jennings Bryan
18	lutesbeveryelcartnohiamsttirefwocendnmepelpo etnioeqlauitniyouosrciyteweslaocdnoemhtnemi otneauqliiytnoerucomony	let us be very clear on this matter if we condemn people to inequality in our society we also condemn them to inequality in our economy	Lyndon B. Johnson
19	uplbethvaaehicasitnnicerulabtitysiowovenokhtn iyerectpxgeitwsawhkhontorjguonwiilmsarncsoi ncotfihoushdvansartdaginnailmesbatihkeppiluss ehritesnasdmde	the public have an insatiable curiosity to know everything except what is worth knowing journalism conscious of this and having tradesman like habits supplies their demands	Oscar Wilde
20	artehekwroetfsiondsuilmproerscdoenspttiongk whoetfsiondtdgsoorehaeresespoivsesielmspuhis cwhaaatimiicrquernrgonntiaiaigvprdotoegcastth boatnneaersheedsthrtceenihientelmspuproofpny earteedrthecarreovaetitnrscovtriuclpeuimiwshes aicmhagitnbrtiinngoeowthmorrldanagkiblvaaiu olrefkhseetgoifndnsoiodtcwhhisehiervrniopdaa ncysonsopnieoss	there are two kinds of impulses corresponding to the two kinds of goods there are possessive impulses which aim at acquiring or retaining private goods that cannot be shared these center in the impulse of property and there are creative or constructive impulses which aim at bringing into the world or making available for use the kind of goods in which there is no privacy and no possession	Bertrand Russell

Appendix III: Samples of Playfair cryptograms and their valid decrypted texts.

#	Cryptograms	Decrypted texts (produced by the software)	Source
1	byntlbneonnuimmzqnhpbkxnqmfqoqnmugclqmeuersuqpnzigqbqyipilqtku	cats are intended to teach us that not exerything in nature ha o a purpose	Garrison Keillor
2	kuinbrnuikcnqmhuvgtnnmybkgbromruknqmmnknqdmpvgniignkoneumokgpgxytqsu	experience is the worst teacher it gives the test before presenting the lesson	Vernon Law
3	rbdpkiommndnrtunhfmgdnqbyuinonebseeuerabonnqoyqvqmeukortuqtqeptigitnpqmzkghdmnnmcv	i refect get it done make it happen thinking i want to slow things down so i understand them better	Jerry Brown
4	qmginetkvgkiigirqftiqyndtyybndqoeuiheuhlpdcoodfqmnivksibrugiqmpqfqnuqmblnmkukiznruku	there is more credit and satisfaction in being a first rate truck driver than a tenth rate eecutive	Bertie C. Forbes
5	qmgpcegpeukntqpyqmeuhkylinueoecneuugcogpflgvlndornqpfonrntimncblcrprsfpdqporknpdcoofba	the likeliness of a thing happening is inversely proportional to its desirability fin agles first law	American Cryptogram Association
6	dncnonqmhiviimottlnqbffsqbzmbrugegtefbblqmgidncnigogmnvlgipktngibflsdlhpgvdztgvggvxz	it isn t the burdens of today that drive men mad rather it is regret over yesterday or fear of tomorrow	Robert Hastings
7	qogndzkgpdcoqftigttndelndopqszqmeukoltbldbdnrdnqpglbueqkqvnqoykunipuimnmdsmifqqmmnkgfqgi	one of the first and most important things for a critic to learn is how to sleep undetected at the theater	William Archer
8	pqlbbkphvrvocnndtnkgihtnewqcpqbfspvqhfsipqqbugqmgnfrgiqkghmntnkggtigeunmigtnmikgcneukgcv	an archaeologist is the best husband any woman can have the older she gets the more interested he is in her	Agatha Christie
9	fhcrndqopqvlgihftngieumrknrbmnqrgurpdpenicpkunedknaqrpgprueuhltihfimbririxrvxncaldentiqxkgtihpcz	ambition an overmastering desire to be vilified by enemies while living and made ridiculous by friends when dead	Ambrose Bierce
10	bkbptkpbqytnivdsvigndzkgikonblpogilvxnxctngeltdgmicaqmkigvxyncoevlgitlpdihcoldtgneqmgincimnqqmgnqmgi	chiasma is a structure of the central nervous system formed by the crossing over of fibers from either side to the other	American Cryptogram Association
11	qmghnugiyadnlsplbrsikndpqmkpuhigtnbryuinrtdzkgrbipilpgpqidyqnoyhmieunqqmhuvgfrvqvpirkzegirfqgpcxeumsqnkgrvuhtnblsetltqdrmncy	the generality of princes if they were stripped of their purple and cast naked into the world would immediately sink to the lowest rank of society	American Cryptogram Association
12	izbroeqohukugmigikonafqznusfqznlgiikontllivoblktqocbdncnmqgpgucnrnqugiginihpnttlqmknmnuhonfsvqinbdnuqzgiginihpnttliginfqqytitqqo	during one week recently twenty two percent of programs on british television were repeats of these twenty two percent were repeats of repeats and so on	American Cryptogram Association
13	rsoeblnzayndqoxcnvugrtundnqmgpsrfqrntpvgqmcnbybkcnyqltyvrvxqoqdomadpsfqmigmngbkumugiqoeumnvqqohuknnqunewtiigmzebdokunuqmrbsfnehknmremqueunumgv	congratulations you ve done it the location for this cach is as follows north fifty three three zero nine two one we stone hundred thirteen thirty eight eight nine zero	Geocache Puzzle
14	qmniayfppbbdeikgbrqyzngqbrummdvggegihgsipxnkeuqmhpdgaczndpqmgineoxgifkxikbqbsigndtknqyhgqcneoeeunmbdninmmzkggaaxvgrtquebbkqmkieikgconwlbneqcyqmiqknvfrihbkpqhgrtdznuqmcnhfmgtnkgblieczimdraegiegoncapupwqmvgdumiincoqotkvgmidppdixfoyafoqbsdpqihgqinmdvgeuqmcnsfintldraegi	the playfair cipher is authorized for emergency use in the army but if there is very much chance of messages being intercepted the key word on which the cipher square is based should be changed often this makes the rapid decipher ment by unauthorized persons more difficult all that can be hoped for in this type of cipher	Helen Fouche Gaines
15	nugelsznlntnqbqckunuibrunuhbksnqebtkpbopeugnldkncnqfsigkcnbrhkzdayseqbqckunuimpduenmafrvbynmbfodqlirpudsrntprukunehkntgunubfqkiabrtquncobyfnvimiihrvoenqqmrbicfqqfprqonczsebtppqodfskgcxqfnmqmfqqmnecokiqoicfqqfprqocngqfreuhlprunaqrdeayqnktqugbgrpyocuumgvpdugbfqkfqkgfyyoqtqfnmqmfqqmkpubyvarginetpvgikmztryfcaeupyonclpqbfdorppgclldtgqmkuhptn	enemy outpost has been driven back to its main line of resistance its right flank has been definitely located at road junction five eight seven dash b prisoners captured belong to third battalion sixth infantry they state that their second battalion is holding a line which passes over hill six zero five dash a they also state that they will be reinforced today by infantry and artillery from the east	William F. Friedman

Appendix IV: Samples of Arabic substitution cryptograms and their valid decrypted texts.

#	Cryptograms	Decrypted texts (produced by the software)	Source
1	بسفذإرللدإهلآإرلهفآإ فرإغسفذدإشغءغإهلآإرلثئء	بحمد الله على العمى كما يحمده غيري على الفقر	أبو العلاء المعري
2	آثرضظثآظتيعاظيعظآثيضةظضظثوعؤظتيعاظيعظآثءتآه	السوء لا يمنع من الموت و لأنه يمنع من الحياة	نجيب محفوظ
3	جيءقجعظزضنيءظيؤصزبظؤحظجيجرضؤجقجضظأجيدحضجك ظأجيمجبقجض	العراق يتطلع لمزيد من الاستثمارات والإنتاج والصادرات	جريدة المدى
4	غمةصكيحقأةذذغنغمتمةجصذغنغمصيةهيعةصذنؤةنؤةعهذةغ نغمةجغنغمنةضكمن	من استغفر للمؤمنين والمؤمنات كتب الله له بكل مؤمن ومؤمنة حسنة	محمد بن عبد الله صلى الله عليه وسلم
5	قحقغبأقثقتبقحثخأصهبثصبقحذءتذإبتضطعبوح بوأؤقمبقحزذغصغبطتأطصإؤذ	الانحرافات الفكرية في المجتمع تقود إلى إرهاب الآمنين وترويعهم	جريدة الحياة
6	يدث يزأكديكةبذسكضيدطح وحأكيدثسوسأكويدةجسحكديكسعوذك ينئيكددقهطد	الحداثة لا تعني بالضرورة الحيوية والتغير لا يكون دائما للأفضل	حسن فتحي
7	عالئددآلدئلثاحقلعدطل ظقأبلظإبظئدأبلظدأالثاحقلعدطلودظبأبلظإؤبئدأب	إن الله لا ينظر إلى صوركم وأموالكم ولكن ينظر إلى قلوبكم وأعمالكم	محمد بن عبد الله صلى الله عليه وسلم
8	جحءفئاثصئجاوإاجغآظجثائآ مثاغءا سغجحاطءجاءةءا ءءجاغحتاشإجصاحهةباجثآطجصوج	الندم ربما هو اعتذار متأخر عن أفعال كنا نظن أننا على صواب لحظة ارتكابها	فراس الهكار
9	هظوطظغميذوخسيءوظيهظبسديوهظغسيقطيءوظيهظبسديشطظ سيهظهخذبهكيحسيغشهئيقطيذف سإئ	العولمة تعني جعل الشيخ عالمي أو جعل الشيخ دولي الانتشار في مداه أو تطبيقه	ويكبيديا
10	كلرخيؤشئألةئكلضعجبئنظأبئتيكأكضئىءضهشئؤذخإضكءئسخئض نهزئظنلئكلىءيتشئدلةئكلضءنخأ	السيطرة على التضخم ودعم قطاعات منتجة ركيزتان في توجه دول المنطقة إلى التنويع	جريدة الحياة
11	يءصويحسوسدفذصءويديءجض ندصسدذجوصدفرفنه مصدخظنينيفدآيءصدوجبهديءشعخديصفءبفدذفاديءأطن	المواطنون تحملوا الكثير من حكومة تفترسهم بقرارات ظالمة وكأس الغضب امتلأت حتى العصر	القدس العربي
12	بضغنوثسطئغنو قنبزفقغبؤغنوأطبغنوجكبوغ قغنوقنطغوبقذ غبؤغنوأقضنغبضغنوصبرنظغنوجفوؤغ قغشإهغنوصرب	في الآخرة المنافقون فى الدرك الأسفل من النار لكنهم فى الدنيا في الصفحات الأولى من بعض الصحف	جلال عامر
13	ذلس خظوختز لوظزى يلذى ظآك ظخوطفذ وك خظوختز دمإخ شينكخ خظئهفف خظإخئز ظخذن وك خظلدنى ظظوختز ظآبقط دئعخدجط	بعث الماضي عملية صعبة لكن لامهرب من الماضي وإذا قصدنا التحرر الذاتي لابد من العودة للماضي لكشفه وتجاوزه	هشام شرابي
14	ؤهذوؤغثاةدعثواثؤهظنكظتثؤهصةكجعؤعاثجزدءمثأءعامثجظتذبث فهظؤثيؤعثؤهصةكجعؤعثدؤهق أثتهبثؤهظنكظتثفهظؤثيؤعثؤهذوؤغثدؤهتفةثزإاإ	النفاق يسود في المجتمع الإستبدادي بصورة طردية بمعنى كلمازاد الإستبداد والضغط على المجتمع كلمازاد النفاق والعكس صحيح	علي الوردي
15	هصشيغخويهةكخيئهطيشهضيضظبديكشهضيوخخقيخويغ ذميكخويإبوكبميكخويغةشوثيكخويغهيكاهيعزثينخجيوخأنوذميو خغزخوإهميغكيوخسهضهميكخويوخبوسميوخثكةهكضيكخويةاج يلهذوضد	يجب ألا يحول شيء بين نفسك وبين الله لا أئمة ولا قساوسة ولا أحبار ولا أي وصي آخر على الزعامة الأخلاقية أو الدينية ولا السادة الروحيون ولا حتى إيمانك	شمس الدين التبريزي
16	ضرضذجةغآذجدذآثادذتؤكغضذضعإهذؤدذضخضيكضلذضخآكذ ؤخآزضذ كذضخءضسكذاذتؤخآثذآيؤةذعضختؤضغبذاذثةةذد تذضخضتعضعذتآإئخذؤخحذد تذضخدآكنب	اذا أردت أن تكون سعيدا ابحث عن الاشياء التي فعلتها في الماضي و جعلتك تشعر بالسعادة و كرر نفس الاسباب ستحصل على نفس النتيجة	إبراهيم الفقي
17	لح رمؤعإم همتحر كخغم لوك لح حجغ همة كخغم آذضى آخظرحإ ورذكسرآ ؤح غرلذرحإ لورصمى رمجترى غذم كآرصتمإر ظوححر طمذ ؤح آة رملوك كعروتر آة رمآخظرح حنظ آتإ هشرفحر	من الأسهل علينا تقبل موت من نحب على تقبل فكرة فقدانه واكتشاف أن بامكانه مواصلة الحياة بكل تفاصيلها دوننا ذلك أن فى الموت تساويا فى الفقدان نجد فيه عطاؤنا	أحلام مستغانمي
18	تضأءشرسأهنأءشنسأثخآطهأكرخأنخءهخأرنقظنثأنخيرحأسءهنأك رخأنخءهخأسآضةآأنخؤشئفأرءهنأنقظنثأنخيرحأسآدوأثضأنخجزأء خنأرتضأنخوضآنأزوأقرخقأهوظئفأرنخؤشئفأزوأءزظخقأهزظخفأرخط خأرنجوفأهضيهنأظضرضأسطرضرنأهضأءظضن أنخؤشئفأرخنأقطرضرنأهضأءظضن أنخوضآنأستضأنخأرهأثهخأرخنأجةنظأرنخؤشئفأجةنظأرخنأثهخ	إن أخوف ما أخاف عليكم طول الأمل واتباع الهوى فأما طول الأمل فينسي الآخرة وأما اتباع الهوى فيصد عن الحق ألا وإن الدنيا قد تولت مدبرة والآخرة قد أقبلت مقبلة ولكل واحدة منهما بنون فكونوا من أبناء الآخرة ولا تكونوا من أبناء الدنيا فإن اليوم عمل ولا حساب والآخرة حساب ولا عمل	علي بن ابي طالب عليه السلام

Appendix V: Samples of Arabic transposition cryptograms and their valid decrypted texts.

#	Cryptograms	Decrypted texts (produced by the software)	Source
1	اسنالمفاونخليذفذلل	الناس من خوف الذل في ذل	علي بن ابي طالب عليه السلام
2	نمهيىولاونفرلاونيروهها	من يهوى النور فالنور يهواه	جبران خليل جبران
3	قصدننكميوهاكوعدكغمنخكراأ	صديقك من نهاك وعدوك من أغراك	علي بن ابي طالب عليه السلام
4	فللاوخامنماعنيولمكتولنعهمميناياةحلن	الخوف لا يمنع من الموت ولكنه يمنع من الحياة	نجيب محفوظ
5	ااافنرستةللزهاأجلأبةيضمنيطعدتنمعألنلجمر	استنفار الأجهزة الأمنية لضبط معتدين على رجل أمن	جريدة الحياة
6	عمإباينجاشابباللعبعليلإاحلطبوإرذلومفقتلندأاملا	إجماع بين الشباب العربي على الإحباط والتذمر وفقدان الأمل	العرب اللندنية
7	لصماحاةلشصخلةيهيامئداصلاراةختتيلحمطتلهيعأوقاملبائدب	المصلحة الشخصية هي دائما الصخرة التي تتحطم عليها أقوى المبادئ	توفيق الحكيم
8	لسراعأدبقعالايلنلدإدتدلوالىنسانصوطاايلعخرالىإكبلجفرارو	رأس العقل بعد الدين التودد إلى الناس واصطناع الخير إلى كل بر وفاجر	محمد بن عبد الله صلى الله عليه وسلم
9	أنكميكاأتكننتكقحمناعبيوضيحقتنتلفيكفنيلسأمنأكردأنيايتحنتر	ك أن يمتك أنك كنت حقا من بعضي وحين قتلتك في نفسي لم أكن أدري أني انتحرت	غادة السمان
10	شكتفردساجةيدةدناإاجابألفطلامينكناطيليمعارإلسنناصخصوعادنلاجرلا	كشفت دراسة جديدة ان إنجاب الأطفال يمكن ان يطيل عمر الإنسان خصوصا عند الرجال	جريدة الحياة
11	نمينكأكتاردهللسماخنللكلشكءيخلشوصذيفلااهكأنوجنولولاداهلليرحنافيصلدسمفأوجينلاةيسكخ	يمكنك أن تدرس الله من خلال كل شيء وكل شخص في هذا الكون لأن وجود الله لا ينحصر في المسجد أو في الكنيسة	شمس الدين التبريزي
12	كهنلنامضايلوررنأقتنفردلمنكركاكانيفحلجبةنأنققابيللقلانلومنلعمتسطلقموأهوخاأقفنقفخيلوه	هل كان من الضروري أن نفترق لندرك كم كنا في حاجة لأن نبقى قليلا لنقول ما لم نستطع قوله أو ما أخفقنا في قوله	واسيني الأعرج
13	القكدنوتادللثةاعلموانةياملمتجعمالثعانيتحثنمايناراتاقلندالاسسعشيرطليااناعنطاببواجدونوسجماتفوانزياالظنمثوامللياللعا	لقد كانت الدولة العثمانية والمجتمع العثماني حتى ثمانينات القرن السادس عشر يعطيان الانطباع بوجود انسجام وتوازن في النظام والمثل العليا	خليل اينالجيك
14	القولتذلركايوتاللاميضلااعململتيكذنربادعسونياسانابرلمغنملذسكظنمللعيقهنانيفذهاهحلايمةشتثبنيابالنمواحاالنمواناسكرانتااخلصاخة	لا وقتل لذكريات ولا الماضي العالم لم يتذكرنا بعد وسينس انا بالرغم من ذلك سنظل معلقين هنا في هذه الحياة متشبثين بالامنا واحلامنا وانكسارتنا الخاصة	أماني العنزي
15	إجأركبنرينهكيمملأنأانسيأنهربتكياكنئمنااكايهنيأنعتلدوولقلحعهضأنظفايآخرلنوهخءطاامهتايلتيشكيحفقهلاللجعودوىحوامبهسهيبلتتيارراهطأخلءهمجئراوه	إن أكبر جريمه يمكن لأي أنسان أن يرتكبها كائنا من كان هي أن يعتقد ولو للحظه أن ضعف الآخريين واخطاءهم هي التي تشكل حقه في الوجود على حسابهم وهي التي تبرر له أخطاءه وجرائمه	غسان كنفاني
16	هنميلاتوكنياياضبناليادسويناهنمرعوتكجبنابييمهايلنكتانيسملمونحيييمسيوبدهوييعشذيلياخالينانهمنعموتكاموضارياملوانفيميهنكتاقطناساونعبتانبلناالريبةنسابلاتدأرضاقان	لا يهمني ان تكونا بيضا سود لا يهمني ان تكون عربي اجنبي لا يهمني ان تكون مسلم مسيحي يهودي بوذي شيعي الخ لا يهمني ان تكون معارضا وموالي ما يهمني فقط ان تكون انسان باعتبار الانسانية بدأت بالانقراض	نيسان خالد
17	ذدداغبرجالكلمالداحبليقفيابعرلاتضابنلدمالكلخلةانيلادةلالرةسآتبخليتبوقللاثأستاوعلباتراكلووقمالهتنابنةاكذيةفزرلرةكاتخرياتدمالكلسلةانيلرةحابنتغيتألااهباواءبدءاعرشلتحثايحامقستلمايربكايغنأنصقموهتخمدهئا	بغداد ذلك الجرح الدامي في قلب العرب النابض تلك المدينة الخالدة الآسرة التي خلبت القلوب واستأثرت بالعقول وكانت لها مكانة بارزة في ذاكرة التاريخ تلك المدينة الساحرة التي تغنى بها الأدباء والشعراء حيث احتلت قسما كبيرا من أغنياتهم وقصائدهم	طه الراوي
18	علوياجهاارلاطأخمألقنلاضريحنجلسااويمةحالننلنااتواوتابواتتاعهجبصاأشبالمهاتإرلكشباماسبلنلإاملمناخلناهامتسارموخلةطئالابنانهيئايصجالداجورتلااشمغرلراياضورعببعمتكلائناناانوحيلابتانالوضإاتلباتلافابإلةنمغتلرايلتحالوايييئثويلباةمسبزبأبسالابحتاخيلارراح	يواجه العالم أخطر انقراض للأجناس الحية من الحيوانات والنباتات وتوجه أصابع الاتهام بشكل مباشر إلى الإنسان من خلال ممارساته الخاطئة ومن بينها الصيد الجائر والاتجار غير المشروع ببعض الكائنات من الحيوانات والنباتات بالإضافة إلى التغير المناخي والتلوث البيئي بسبب أزمة الاحتباس الحراري	العرب اللندنية

Appendix VI: Samples of Arabic Playfair cryptograms and their valid decrypted texts.

#	Cryptograms	Decrypted texts (produced by the software)	Source
1	رهبإهغلمنهلتفغتآلتإأفنتلفئذفظتآنلشاهملوألترتبلإهفإشاغإطصفهربلتأغ شهشعاخفههلهشكشثالتإنإللشثظصمهلغإأظساكفلثذجثنلرزكذوماشاه	ثمة نوعان من الشقاء الأول ألاتحصل على ماتتمناه والثانى أن يأتيك وقد تأخر الوقت وتغيرت أنت وتغيرت الأمنيات بعد أن تكون قدشفيت بسببها بضع سنوات	أحلام مستغانمي
2	تلأغشتبظطخالشهتللتآركخلتطتاناناشضظبملرطزؤغلإدشملرلابطونهظ بغضبنانظتكفهوملياتزمتفبالآإشفثجلإرلتلنهشمفرتلنهشملمغضردلهرآلتد ريضئ	لا وقتل لذكريات ولا الماضي العالم لم يتذكرنا بعد وسينس انا بالرغم من ذلك سنظل معلقين هنا في هذه الحياة متشبثين بالامنا واحلامنا وانكسارتنا الخاصة	أماني العنزي
3	نضرملتخظئصذرلتذاأتمشبذثملمظتءظشنآرفندجردلتنيإتئيظفتنآرلتذاا تظتئبتلإؤلتغزتيتلرذإهفكنشيظنزتطانتلفخذفنيلرلتسأحظفتننزرآبتدلطإ	إذا طالبك أحد بالدليل والبرهان على علوم الأسرار الإلهية فقل لهما الدليل على حلاوة العسل فلا بد أن يقول لك هذا علم لا يحصل إلا بالذوق فقل له هذا مثل ذاك	ابن العربي
4	كهعذاديسخيأغةيعغؤقآسطنكههوجاًعةخيأغةيعغؤقآسكةخأكةءإمحم ةةسهكتثظزيبيخظخشهمةؤنيخوثفدنؤإبأثءلسفهشكاخيأغةيترقظوثف دغعجآعغبقإقبقدشهجآعغبخبكاإبقشهتآدبثك	لايهمني ان تكون ابيض اسود لايهمني ان تكون عربي اجنبي لايهمني ان تكون مسلم مسيحي يهودي بوذي شيعي الخ لايهمني ان تكون معارض او موالي ما يهمني فقط ان تكون انسان باعتبار الانسانية بدأت بالانقراض	نيسان خالد
5	هفنهئنإلهفلخطزنأشانهلإقحإلإشزشربروإظشلرلشالتسقاخآيانشهيدكءآر يامتنهجمرفينبيفدمشفذصطمشفبإإؤإلتفثوديظتفتةجصهأدففبؤساإانفنهو ظضيظتلظهجطخياالتلمشيتطثن	أتمنى أن يأتي بعد موتي من ينصفني ويستخرج من كتاباتي الصغيرة المتوازعة ما فيها من روح الإخلاص والصدق والحمية والتحمس لكل شيء حسن وصالح وجميل لأنه كذلك لا عن رغبة في الانتفاع به	مي زيادة
6	جطآانيغتلشلتلعفدفضاؤإلتشراهتطخيلتظخاخيخإطملفظنشجنفطكتآيم ذنشلتطفخآلتغؤطرنثصماهيثغمخجتخلمجأودفعذبهارسضامشظخلتقز جمشهاحيامتتشظإنشؤسالبفنشرضاضآانانفلظيلتطصاإآإ	رغم المشكلات الاقتصادية والاجتماعية الكبيرة يمكن القول بواقعية ان دول القارة السوداء بدأت تصطوحثيثا نحو مستقبل متجدد يواكب العصر وتوفر فيه التكنولوجيا حلولا جديدة للمشاكل القديمة	جريدة الصباح
7	دفثالتلمشحردلشمشتنرولشلتمخمتخلآخإشطرجماضلففتزآتطشارطالل تإندربظشخظتمتاهاثشرطهظهفزطدلتظهشملإلتحآشضخيياالتفعرداخلت ةضرلخاؤضمشتهبللتودشهآلانبففإمشاذنشبإبذكزلتأغشيالتظدرطأ	صارت الانفجارات والهجمات الإرهابية رويدا رويدا شيئا اعتياديا الأمر الذي جعلها تتراجع عن تصدر العناوين الرئيسية في التقارير الإخبارية سواء على المستوى المحلي أو الدولي ولبعض الوقت في العراق	القدس العربي
8	فدفبمشلتثهتاإذفجأآلتثهتاناأنيليلتثوفنيصتلثخجهشميشجؤتطملفنذج لمرلتحيخييالتنهتمرشتشظتاإخيلتشاشثظتلتهاظتاإأللآحلمشمربليلتشض اذتاإهركئثفئلفإهوإازشكاغههإالثهانمشاأإإظإإهايطخمشيصتلظهإهلخيع جمفمإشثيثطمش	اصلحوا التعليم أصلحوا التعليم وإلا فالجهل أفضل ابحثوا وفتشوا عن الأسباب الخفية في المناهج التعليمية التي تجعل المتعلمين فئرانا وخرافا ليس لديهم ثقة حتى في أنهم يستطيعون أن يتعلموا أو يمكن أن يفكروا فضلا عن أن يبتكروا أو يخترعوا	أيمن صبري فرج
9	عخخاآلتتماهرآلشلتشإإائيالبذاشمشتظنامشيذوؤمرتشاشرحأمتشءضا ظيمتنياآلتشهشجملتظفطفاذفاًإإظآرشءجبظتؤبمسآيبأنأتزاوثإاهءزنلتغه رآتشفشاهاثإخدتاذآروؤاهشزرنحلخكقتلتدلطخؤإلتثهخدآدرديمبلطلمثآيد بضشلملتأأشرآأ	كثرة الاهتمامات اليومية في البيت والعمل والضغوط التي ترافقها تؤدي كلها إلى التوتر والقلق الدائمين كما تؤثر على جودة النوم ومدته جميع هذه العوامل تش تت تركيز الدماغ وت تسب ب في ضعف الذاكرة والتعرض مرلرا إلى ظاهرة النسيان المفاجئ	العرب الندنية
10	مهملأمبنهبآلتصاأمبنهذفنيتإأمبنهذبعوتأأشلإشموأفختصملأوأنرتانظ اكفكشثظإذإإبملتبقؤإحذفأبمرآيشأجءكتامزأنشمفبكبلتفهنالتدفيافعمسم لنيتإمشتظانفكمسملنيتإأندرظحلهلمشصملنهلتطربيفعمسملنيتإنجدفآ حملثءضانيتإ	إننا وإن لم نرى الحق وإن لم نصل إليه وإن لم نبلغه فهو فينا وهو يحفزنا وهو مثال مطلق لا يغيب عن ضميرنا لحظة وبصائرنا مفتوحة عليه دوما ولحظة التأمل الصافي تقودنا إليه والعلم يقودنا إليه ومراقبتنا لأنفسنا من الداخل تقودنا إليه وبصائرنا تهدي إليه	مصطفى محمود
11	نهلتصأفآطفانذشنرمتإهلتلمدشهوإغفآملثشارآكتتلتكافرناحزمذبازتقوعخا عثمتبللتكتاوغقاخونهاإذبتآبوزوفنهوطتتسرناحزمظهلتاأشتآتخشرلشوالدي طلملتشراهتطلتلمدشإلدخمجاتينوذلمظتآلفذكتطرانرطايوأياألدشنعفد خمجاآإظهإشاإ	من الحقائق المسلم بها أن الانسان هو كائن اجتماعي لا يعيش بمفرده بل تدفعه غريزته الى العيش مع غيره من أبناء جنسه لأنه عاجز بمفرده عن الوفاء بحاجاته واذا كان الاجتماع الانساني ضروري للإنسان على الصعيد المادي فهو في نفس الوقت ضرورة معنوية	جريدة الصباح
12	دحفهأغهإملفظأنياذبرطملظتؤليطيبمشرصئيتإدللتضذضمآرإهاإظهمشيالد تأضطشآتانرضخمدلطيهلصقاتآيانيعانهلتنإجهدرشلبذركئظوخيأمآرإها شرلجمليالانفلخطئياليظتلنرشدازطتؤهبثآرثلنشايخقلالشأإغألتاهكخيا ألدشتلشثتمنيلبشصلتهطبل	صرت أوقن أن القوم في بلادنا على شاكلة واحدة في هذا الصدد إما أن يمعنوا في الإصغاء المخدر إذا كانت حصيلة المتكلم من المأثورات البراقة غنية وإما أن يتباروا في المشاركة في الكلام باستدعاء وربما بتوليف حكايات وأقوال تمضي في نفس الاتجاه إلى نفس المعنى	خيري شلبي
13	تلظتفطبياظفآيئرلاتزنحنمشاذلإيلتشقزذباذلإوأثهذحتظكفضاإأبقئأوهشقز ذبتنمشتشقزذبغمفآتأهشقزذبجلدآهوآرانآوهأنهونكشمرانآوهنهوآريباخأنهو آرتفخادآهوآرشنظمشإنهونلتطهاتأوظضشملتثهذحشبآركءجلخابخخقآر شنتلضاآثندللتهطبلوأهأطهملشبتمتائيلإطبلتوضتلآرظفآياإطلودالضشر اهتطشإتاصذلإفمهشضامليقختدآملاظفآياوظاخهاضاإلهن	لا علاقة للطائفة بالمذهب أو الدين فالتعصب للدين هو تعصب لعقيدة أو لنص أو هو تعصب لله والتعصب لطوائف هو تعصب لبشر منهم المؤمن ومنهم غير المؤمن منهم الخير ومنهم الشرير منهم الوطني ومنهم العميل وهكذا والتعصب لجماعة بشرية بحكم الولادة بهذا المعنى هو نوع من الجاهلية في نظر الإسلام الطائفية نظام سياسي اجتماعي وليس دينا أو تدينا فكثير من الطائفين غير متدينين	عزمي بشارة