Novel Reversible Text Data De-Identification Techniques based on Native Data Structures

BY

Bayan Al-Abdullah

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy at the University of Sussex

School of Engineering and Informatics

Department of Informatics

University of Sussex

Brighton

BN1 9QT

December 2021

Declaration

The work described in this thesis, carried out in the school of Engineering and Informatics, I hereby declare that this thesis has not been and will not be submitted in whole or in part to another University for the award of any other degree.

Signature: _____

Bayan Al-Abdullah

Acknowledgements

This brief page will never be able to adequately express my gratitude.

I would like to express my deepest gratitude to my supervisors, Dr. Natalia Beloff and Prof. Martin White, for their efforts in providing me invaluable guidance, encouragement, motivation, and continuous support during this work. I was fortunate to have supervisors concerned about my work and who responded to my questions and queries so promptly.

None of the work in this thesis would have been possible without the support of my husband, Abdullah, who was willing to put his life on hold for years while I pursued something I considered ‘’really important.” My parents, who have been an endless source of great assistance and love to complete my PhD. My kids; Emad, Alhanouf and Reema, whose patience and love inspired me to achieve my goal, a special thanks for them for tolerating the long hours being away from them working on my research.

My sincere thanks to Princess Nourah bint Abdulrahman University and the Saudi Arabian Cultural Bureau for giving me the opportunity to complete my PhD study and support me financially and morally.

From the outside, a PhD appears to be an impressive individual achievement. From my own experience, the accomplishment that I have had at least as much to do with the people that I have been fortunate enough to become involved with than any talent that I have had.

UNIVERSITY OF SUSSEX

Submitted for the degree of Doctor of Philosophy

Novel Reversible Text Data D-Identification Techniques based on Native Data Structures

Abstract

Technological development in today's digital world has resulted in the collection and storage of large amounts of personal data. These data enable both direct services and non-direct activities, known as secondary use. The secondary use of data can improve decision-making, service experiences, and healthcare systems. However, the widespread reuse of personal data raises significant privacy and policy issues, especially for health-related information; these data may contain sensitive data, leading to privacy breaches if compromised. Legal systems establish laws to protect the privacy of personal data disclosed for secondary use. A well-known example is the General Data Protection Regulation (GDPR), which outlines a specific set of rules for sharing and storing personal data to protect individual privacy. The GDPR explicitly points to data de-identification, especially pseudonymization, as one measure that can help meet the requirements for the processing of personal data.

The literature on privacy preservation approaches has largely been developed in the field of data anonymization, where personal data are irreversibly removed or obfuscated and there is no means by which to recover an individual's identity if needed. By contrast, pseudonymization is a promising technique to protect privacy while enabling the recovery of de-identified data. Significantly, many existing approaches for pseudonymization were developed long before the GDPR requirements were established, and so they may fail to satisfy its provisions. Therefore, it is worthwhile to offer technical solutions to preserve privacy while supporting the legitimate use of data.

This thesis proposes a novel de-identification system for unstructured textual data, known as ARTPHIL, that generates de-identified data in compliance with the GDPR requirement for strong pseudonymization. The system was evaluated using 2014 i2b2 testing data. The proposed system achieved a recall of 96.93% in terms of detecting and encrypting personal health information, as specified under guidelines provided by the Health Insurance Portability and Accountability Act (HIPAA). The system used a novel and lightweight cryptography algorithm E-ART to encrypt personal data cost-effectively and without compromising security. The main novelty of the E-ART algorithm is the use of the reflection property of a balanced binary tree data structure as substitution method instead of complex and multiple iterations. The performance and security of the proposed algorithm were compared to two symmetric encryption algorithms: The Advanced Encryption Standard and Data Encryption Standard. The security analysis showed comparable results, but the performance analysis indicated that E-ART had the shortest ciphertext and running time with comparable memory usage, which indicates the feasibility of using ARTPHIL for delay-sensitive or data-intensive applications

List of Publications

Conference Paper

- Alabdullah, Bayan, Beloff, Natalia and White, Martin (2018) ***Rise of big data – issues and challenges.*** SCS-NCC' 2018 Saudi section 21st Saudi Computer Society National Computer Conference, Riyadh, Saudi Arabia, April 25-26, 2018. Published in: 2018 21st Saudi Computer Society National Computer Conference (NCC). 1-6. ISBN 9781538641118

- Alabdullah, B., Beloff, N., & White, M. (2021) *ARTPHIL: **Reversible de-identification of free-text using an integrated model***. EAI SPNCE 2021 - 4th EAI International Conference on Security and Privacy in New Computing Environments. (Accepted EAI SPNCE 2021 - 4th EAI International Conference on Security and Privacy in New Computing Environments December 10-11, 2021)

Journal Paper

- Alabdullah, Bayan, Beloff, Natalia and White, Martin (2021) *E-ART: **a new encryption algorithm based on the reflection of binary search tree***. Cryptography, 5 (1). a4 1-15. ISSN 2410-387X

List of Abbreviations

E-ART	Encryption through ASCII Reflection Tree
GDPR	General Data Protection Regulation
ASCII	American Standard Code of Information Interchange
PHI	Personal Health Information
DES	Data Encryption Standard
AES	Advanced Encryption Standard
BIC	Bit Independence Criteria
HIPAA	Health Insurance Portability and Accountability Act
PPDM	Privacy Preserving Data Mining
PPDP	Privacy Preserving Data Publishing
DSR	Design Science Research
SSN	Social Security Number
DoB	Date of Birth
QID	Quasi Identifier
FHE	Fully Homomorphic Encryption
NIST	National Institute of Standards and Technology
SAC	Strict Avalanche Criterion
LWDC	Lightweight Dynamic Crypto
CSPRNG	Cryptographically Secure Pseudo Random Number Generators
DED	Dynamic Encryption Determination
ARTPHIL	Anonymized Reflection Tree Philter
SMC	Secure multiparty computation

Glossary

Data de-identification	Any process that removes the association between data and the individual with whom the data is initially associated. It required removing or transforming Personal Identifiable Information (PII)
Anonymization	Anonymization is a data processing technique that removes or modifies personally identifiable information; it results in anonymized data.
Linkage attack	A linkage attack is an attempt to re-identify individuals in an anonymized dataset by combining that data with background information. The linking may use quasi-identifiers, such as zip or postcode, gender, salary, and so on that are present in both sets to establish identifying connections.
Pseudonymization	Pseudonymization is a particular type of de-identification in which the names and other information that directly identifies an individual are replaced with pseudonyms.
Pseudonym	A computed or assigned value that is substituted for one or more data elements in data subject's record

Equivalence class	An equivalence class is the name that we give to the subset of S which includes all elements that are equivalent to each other.
Data controller	An entity that, alone or jointly with others, determines how and why personal data are processed
Re-identification	The process by which de-identified personal data is matched with its true owner
Structured data	Structured data consist of clearly defined data types with patterns that make them easily searchable such as relational database or spreadsheet
Unstructured data	Unstructured data is data that is not formed in a predefined manner, such as textual note
Homogeneity attack	Refer to the knowledge gained by correlating quasi-identifiers and sensitive attributes.
Background attack	Refer to the attack in which an adversary has some prior knowledge about the data subject.
Spatial crowdsourcing	Refer to the type of crowdsourcing in which workers complete their assigned tasks at a specific location.

Table of Contents

Chapter 1 : Introduction **1**
1.1 Research Background 1
1.2 Research Motivation 2
1.3 Research Questions 4
1.4 Research Objectives 5
1.5 Contributions 6
1.6 Outline of Thesis Structure 7
Chapter 1: Introduction 7
Chapter 2: Background and Related Work 7
Chapter 3: Methodology 7
Chapter 4: E-ART Design 7
Chapter 5: E-ART Performance and Security Evaluation 8
Chapter 6: ARTPHIL: De-Identification Model Design and Evaluation 8
Chapter 7: Conclusion and Future Works 8
Chapter 2 : Background and Related Work **9**
2.1 Privacy 9
2.1.1 Definition of Privacy 10
2.1.2 Personal Data Classification 12
2.1.3. Privacy Preserving Principle 16
2.1.4. Re-identification Risks 17
2.2 Privacy Preserving Approaches 18
2.2.1 Privacy Preserving Data Publishing 19
2.2.2. Privacy Preserving Data Mining 24
2.3.Data De-identification in GDPR 30
2.3.1. GDPR Principles for Processing Personal Data 30
2.3.2. The Three Types of Data in GDPR 32
2.3.3. Effectiveness of Data Sanitization Techniques to Mitigate Re-identification Risk 34
2.3.4 Obligations Under the GDPR 37
2.4. Cryptography 42

2.4.1 Cryptography Evaluation and Performance Criteria 43
2.4.2. Security against Attacks 46
2.5. Dynamic Key 48
2.6. Lightweight Cryptography Discussion 49
2.7. Summary and Conclusion toward Research Gap 57
Chapter 3 : Methodology 60
3.1. Research Methodology 60
3.2. Design Science Methodology 61
3.2.1. Awareness of the problem 64
3.2.2. Suggestion 64
3.2.3. Development 64
3.2.4. Evaluation 65
3.3. Justification of the Research Method 67
3.4. Rationale for Research Approach 68
3.5. Research Design 69
3.6 Dataset 71
3.7. Summary 73
Chapter 4 : E-ART Design 74
4.1. Algorithm Design 74
4.1.1. Stage 1: Substitution Method 75
4.1.2. Stage 2: Adding offsets 79
4.1.3. Stage 3: Dynamic key 80
4.2. Encryption Process 82
4.3. Decryption Process 86
4.4. Proof of Correctness 88
4.4.1 Empirical Proof 88
4.4.2. Mathematical Proof 90
4.5. Summary 93
Chapter 5 : E-ART Performance and Security Evaluation 95
5.1. Introduction 95
5.2. Experimental Setup 95
5.3. Evaluation Metrics 96
5.3.1. Performance Parameters 96

5.3.2. Security Parameters 97
5.4. Results 100
5.4.1. Performance Analysis 100
5.4.2. Security Analysis 104
5.5. Security against Attacks 111
5.5.1. Brute Force Attack 111
5.5.2. Chosen and Known Plain-text/Cipher-text Attacks 112
5.5.3. Statistical Attacks 112
5.6. Summary 113
Chapter 6 : ARTPHIL De-Identification Model Design and Evaluation 115
6.1. Introduction 115
6.2. Problem Formulation 116
6.3. De-identification of Protected Health Information under HIPAA 118
6.3.1. The HIPAA Expert Determination Method 118
6.3.2. The HIPAA Safe Harbor Method 119
6.4. De-identification as Named Entity Recognition 120
6.4.1. Evaluation Matrices 123
6.5. Stat-of-the-art Solutions 124
6.5.1. Philter 125
6.5.2. DE-identification System based on Artificial Neural Networks(ANNs) 127
6.5.3. DE-identification system based on ANNs and CRF 127
6.6. Proposed Method 127
6.6.1. PHI Detection 127
6.6.2. Replacement strategy 128
6.7. Experiment Setup 129
6.7.1. Implementation 129
6.7.2. Dataset 130
6.8. Results 132
6.8.1. Re-identification Risk 135
6.8.2. Execution Time 136
6.9. Effectiveness of ARTPHIL 137
6.9.1. ARTPHIL and De-identification Systems: 137
6.9.2. ARTPHIL and GDPR requirements: 139

6.10. Summary 140
Chapter 7 : Conclusion and Future Works 142
7.1. Thesis Summary 142
7.2. Suggestions and future direction 146
7.3. Summary 148
References 149
Appendices 163
Appendix I: E-ART Code (Java) 163
Appendix II: E-ART Evaluation 168
Avalanche effect 178
Appendix III: ARTPHIL (Python) 180
Appendix IV: Binary Tree Data Structure 193
Appendix V: Cryptography 194
Advanced Encryption Standard 196
Data Encryption Standard 200
Security against Attacks 203

List of Figure

Figure 2-1: Aspects of Privacy [23] 12
Figure 2-2: Privacy Preserving Approaches 20
Figure 2-3: K-Anonymity [39] 21
Figure 2-4: Laplace Mechanism [49] 25
Figure 2-5: Letter and their Relative Frequency [90] 46
Figure 3-1: Design Science Research Processes and Outcomes [148] 62
Figure 3-2: Research design 70
Figure 4-1: Algorithm Design Process 75
Figure 4-2: Binary Tree of E-ART 77
Figure 4-3: Illustration of the Substitution Method (Stage 1) 78
Figure 4-4: Encryption with Different Keys 83
Figure 4-5: Conceptual Overview of the Encryption Process 85
Figure 4-6: Conceptual Overview of the Decryption Process 88
Figure 4-7: Example before and after encryption by E-ART algorithm of a) the original clinical note, b) the encrypted note, and c) the decrypted note 90
Figure 5-1: Snapshot of Implement Algorithm in NetBeans 96
Figure 5-2: Performance Analysis Process 98
Figure 5-3: Comparison of Memory Consumption of (a) Encryption (b) Decryption for AES, DES and E-ART Algorithm 102
Figure 5-4: Comparison of execution time of (a) Encryption and (b) Decryption for AES, DES, and E-ART algorithms 103
Figure 5-5: Histograms of (a) plain-text, (b) E-ART, (c) AES, and (d) DES 106
Figure 5-6: Snapshot of NIST Test 109
Figure 6-1: The Process of DE-Identification using Philter [149] 126
Figure 6-2: Integration of E-ART with Philter 129
Figure 0-1 Encryption and Decryption Processes in AES [223] 197

Figure 0-2: Data State Array 197
Figure 0-3: Add Round Operation [224] 198
Figure 0-4: Substitute Byte Operation [224] 198
Figure 0-5: Shift Rows Operation [224] 199
Figure 0-6: Mix Columns Operation [224] 200
Figure 0-7: DES Encryption Function [225] 202
Figure 0-8: DES Function *f* [226] 203

List of Table

Table 2-1: Example of Medical Data 14

Table 2-2: Example of Voter Registration Rolls 15

Table 2-3: De-identified data 15

Table 2-4: Comparison of privacy-preserving approaches in terms of the employed privacy preserving principle 28

Table 2-5: Summary of the De-identified Data under GDPR 34

Table 2-6: Robustness of data sanitisation methods against risks 37

Table 2-7: Summary of different types of data and obligations required 41

Table 4-1: ASCII Table 76

Table 4-2: Summary of the Notations 79

Table 5-1: Recommended size of Bit-stream for each NIST test 99

Table 5-2: AES Performance 100

Table 5-3: DES Performance 101

Table 5-4: E-ART Performance 101

Table 5-5: Comparison of file size before and after encryption for AES, DES, and E-ART algorithms 103

Table 5-6: Avalanche Effect Performance 104

Table 5-7: BIC Performance 105

Table 5-8: ASCII Histogram Frequency Value 107

Table 5-9: Statistical Test Results 111

Table 6-1: Comparison of methods used for PHI Detection 123

Table 6-2: Performance of Current State-of-the-art Models 124

Table 6-3: De-identification and Re-generated of Example Clinical Note 130

Table 6-4: PHI Distributions in the i2b2/UTHealth 2014 DE-Identification Corpus 131

Table 6-5: Overall Model Performance 133

Table 6-6: Recall by Tag 133

Table 6-7: Recall by PHI Category 134

Table 6-8: The Probability of Risk Re-identification 135
Table 6-9: The Execution time for E-ART_ Philte 136

Chapter 1 : Introduction

1.1 Research Background

Advances in information technology have enabled organisations to store, share, and analyse a large amount of personal data. This promises to yield new insights into questions that have been difficult or impossible to answer in the past. However, the storage and sharing of potentially sensitive data pose serious privacy concerns. For both legal and ethical reasons, it necessary to preserve individual privacy and confidentiality. Legal systems establish laws to protect the privacy of individual information disclosed for secondary use. A well-known example is the General Data Protection Regulation (GDPR), which outlines a specific set of rules for sharing and storing personal data to protect individual privacy [1]. Data minimisation is one of the fundamental principles of the GDPR, which has strict data retention policies. This protection means that an individual's personal data can be retained for no longer than necessary to carry out the purpose for which the data was initially processed[1] . The GDPR also restricts the use of personal data beyond the purpose for which the data was originally collected (purpose limitations)[2] [1]. However, the new rules and obligations imposed by the GDPR do not prevent the use of personal data for analytics or other useful secondary applications. They do, however, require the implementation of new technical and organisational measures to protect that data. The GDPR explicitly points to data de-identification as a measure that can support meeting the requirements of several of its provisions [2].

[1] (GDPR Article 5(1)(e))
[2] (GDPR Article 5(1)(b))

Data security and privacy issues become even more critical when data is used in healthcare environments, which typically deal with sensitive patient information. In the United States, standards for protecting healthcare information confidentiality were established in HIPAA's Privacy Rule [3]. HIPAA specifies 18 data elements that define Protected Health Information (PHI); these elements must be removed or generalised from data for it to be considered de-identified. Removing all identifiers that fall into one of the 18 elements is said to provide the required anonymity for health data to satisfy the HIPAA data sharing regulations, which allows the data to be used for scientific studies.

This dissertation focuses on privacy preservation through de-identification to reduce the risk of disclosing confidential information associated with collecting, archiving, and transferring personal information about individuals.

1.2 Research Motivation

Previous work on data de-identification and anonymization techniques has been developed within the fields of statistical disclosure control [4], privacy-preserving data publishing (PPDP) [5], and privacy-preserving data mining (PPDM). In these fields, sensitive data are shared with untrusted third parties for secondary use, but information is not disclosed that can be linked to specific individuals. This technique primarily focuses on producing anonymized versions of data by removing, obfuscating, or generalising identifiable personal data. The drawback of these methods is that they are usually irreversible; there is no mechanism by which an individual's identification can be recovered.

In many circumstances, it is essential to refer back to the original data without revealing it to the end-user. This allows the data to be accessed in emergencies or by entities with acceptable levels of access. For example, in PPDM, the knowledge obtained from mining de-identified data cannot be verified from the original data, which might produce knowledge uncertainty [6], [7]. In research platforms, where documents are

shared between different specialists, including scientists, researchers, and physicians, specific cases are usually chosen for further analysis. For instance, in selected cases where specialists conduct a blind diagnosis or annotation procedures, the authorised user can reverse the de-identification process and retrieve the necessary information, leading to a more accurate assessment [8] [9].

Additionally, situations often occur in clinical trials when some of the research subjects need to be approached again for further study [10]. Therefore, reversible de-identification (e.g., encryption and pseudonymization) are often preferred. Several privacy-preserving approaches have been proposed to address the need for reversible de-identification, which involve cryptographic approaches that allow the original data to be requested if needed. For example, Landi and Rao [10] proposed a way to de-identify patient data so that only the owners of the original data or legally empowered entities can re-identify it. Their system uses secure public-key encryption technology to generate a public key based on one or more private keys. Yamac et al [11] proposed privacy preserving solution that combines a multi-level encryption scheme with compressive sensing. The approach has the ability to reverse the De-identification so that an authorized person can recover the degraded part of information using the key. It tried to simplify the key management issue by watermarking the key into the sensed image. However, the computationally expensive decompression might be difficult to apply to big data.

Gulcher et al. [12] also highlighted the need for reversible de-identification to protect genetic research data. The authors proposed an approach for de-identifying biological samples for genetic research based on a third-party encryption method using the Advanced Encryption Standar (AES), a 128-bit symmetric encryption algorithm. This approach would allow later requested access to the research data. However, to enhance security, existing encryption standards rely on larger key sizes and greater numbers of rounds, which can be computationally expensive and challenging to apply to a large amount of data, thus negatively affecting performance. Moreover, low-resourced devices require low

resource implementation to secure data in order to ensure a long battery life of the devices [13]. Therefore, lightweight and practical alternatives must be developed [14]. As a result, there is a need for a new reversible model for de-identifying unstructured data to address the conflicting requirements of preserving privacy while supporting the legitimate use of data.

Based on these considerations, this dissertation focuses mainly on developing a new reversible de-identification model using a novel and lightweight cryptography algorithm. This is expected to address the need for a lightweight de-identification system, thus fulfilling the requirements of data storage and access for primary use by the authenticated user as well as privacy-preserving secondary use.

1.3 Research Questions

This research aims to investigate the problem of privacy preservation for unstructured big data through de-identification. The main research question is

"How to reversibly de-identify unstructured data to address the requirements of preserving privacy while supporting the legitimate use of data?"

This question leads to other sub-questions as follow:

1. What are the issues and challenges with existing privacy-preserving approaches?
2. What are the issues and challenges in existing cryptography approaches?
3. Can the proposed encryption algorithm achieve a trade-off solution between security and performance compared with the Data Encryption Standard (DES) and AES?
4. How can the proposed encryption algorithm be used to de-identify unstructured textual data in order to improve individual privacy?

1.4 Research Objectives

To achieve the research aim, the research question and the issues highlighted in previous sections are addressed in the following objectives:

1. To critically investigate privacy-preserving approaches.
2. To critically investigate cryptography approaches for textual data.
3. To design and develop a lightweight encryption algorithm that can achieve a satisfactory level of security with shorter execution times and fewer resources, thereby meeting the requirement of processing a large amount of data.
4. To evaluate and analyse the developed algorithm in relation to evaluation metrics and benchmark algorithms.
5. To design, develop, and evaluate a reversible de-identification model that uses the proposed lightweight encryption algorithm as a replacement strategy to reversibly de-identify unstructured data.
6. To ensure that the develop de-identification model decouples data elements from re-identifiable linkages to data subjects (by encrypts direct and indirect identifiers). Hence satisfy the requirements for GDPR Article 4(5)[3] compliant pseudonymized data.

The methodological tool used to achieve the research objectives of this thesis is design science research (DSR). This is an approach to scientific knowledge creation that includes the development of novel constructions aimed at solving real-world problems while also making a prescriptive scientific contribution. [12]. The principal outcome of DSR is an artefact that solves a domain problem. There are two artefacts created for this work: first, a novel lightweight encryption algorithm that aims to encrypt textual data in a cost-

[3] GDPR Article 4(5) ("information necessary
to attribute personal data to a specific data subject must be kept separately and subject to technical and organizational measures to ensure that the personal data are not attributed to an identified or identifiable natural person").

effective manner; and second, a reversible de-identification model, , that aims to de-identify health data to reduce the privacy risk for the data subject.

1.5 Contributions

The contributions of this research are summarized as follows:

1. Proposing and developing a novel encryption algorithm, E-ART[4], that overcomes the limitations of existing ciphers in terms of multiple rounds and complex computation, which can undermine performance. The proposed algorithm relies on two key elements: the reflection property of a balanced binary search tree data structure, which is applied to minimize the overhead; and a dynamic key, which promotes a high level of security.
2. Demonstrating the efficiency and security of E-ART by performing empirical comparisons with AES and DSS, two widely used encryption algorithms. E-ART outperformed AES and DES in terms of performance and achieved a comparable level of security
3. Proposing and developing a reversible de-identification system, ARTPHIL [5], to prevent the unauthorized reidentification of unstructured health data. The system combines the state of the art in pattern matching and statistical modelling to extract personal identifiers specified in free-text, and then to encrypt these identifiers using the E-ART algorithm.
4. Evaluating ARTPHIL's performance using the i2b2/UTHealth 2014 de-identification corpus.
5. Helping to bridge the gap between the new GDPR obligations and the legitimate use of data. This contribution arises from the fact that ARTPHIL can be used to produce de-identified data that comply with the GDPR's requirement for strong pseudonymization.

---

[4] bayan6060/eart1 (github.com)
[5] bayan6060/philter_eart (github.com)

1.6 Outline of Thesis Structure

This section presents the order of the thesis chapters and provides a brief description of each chapter

Chapter 1: Introduction

This chapter introduces privacy-preserving approaches and their limitations. It outlines the motivation for the proposed solution, the research questions, the research aim and objectives, and the methodology used to achieve them.

Chapter 2: Background and Related Work

This chapter first reviews related work and presents background information related to the thesis. It first defines privacy key aspects, privacy-preserving criteria, and approaches used to reduce privacy risk for the data subject. Then, it reviews and analyses the new GDPR regulations and obligations for processing personal data. In turn, the chapter introduces background material on the cryptography field, including encryption categories, design principles, and attacks. The chapter closes with a discussion of lightweight cryptography.

Chapter 3: Methodology

This chapter describes the methodology used to achieve the research objectives. Justification of the research methods is presented alongside the rationale for the research approach. Details of how the results are tested and analysed are also included.

Chapter 4: E-ART Design

This chapter describes the design phases for the new E-ART algorithm. The design consists of three phases: Substitution Method, Adding Offsets, and Dynamic Key Phase. The encryption and decryption processes, along with the proof of algorithm correctness, are included in this chapter.

Chapter 5: E-ART Performance and Security Evaluation

This chapter presents the validation of the proposed E-ART cryptographic algorithm through a series of experiments. Comparative performance and security analysis are conducted to demonstrate the effectiveness of the novel design. The performance analysis parameters include processing time, memory consumption, and file size. Security was assessed through the avalanche effect, the bit independence criterion**,** frequency analysis, and the NIST statistical test.

Chapter 6: ARTPHIL: De-Identification Model Design and Evaluation

This chapter reviews information extraction techniques used for the detection of sensitive data (specifically, personal health data). In turn, it presents the development of an integrated model, ARTPHIL, that integrates state-of-the-art tools for extracting personal identifiers from free-text, the purpose being to detect confidential information and encrypt it with the proposed lightweight encryption algorithm E-ART. The chapter also evaluates the proposed model in terms of precision, recall, and F-measure. Processing time and the probability of re-identification risk are also measured.

Chapter 7: Conclusion and Future Works

This chapter presents conclusions arising from the thesis and specifies directions for future research.

Chapter 2 : Background and Related Work

The overall goal of privacy-preserving research is to develop techniques that allow to get the best utility out of a dataset without violating the privacy of the data subject. Different techniques and different regulations to address this problem have been proposed. However, there is no universally suitable technique; instead, it will depend on the specific use case and needs.

This chapter first discuss the privacy concept in section 2.1, privacy preserving approach in section 2.2 de-identification and GDPR in section 2.3. Section 2.4 gives an overview of cryptography approaches and their evaluation criteria. Section 2.5 introduced the concept of dynamic key. Section 2.6 discuss the lightweight cryptography approaches and finally chapter summary and conclusion toward research gap in section 2.7.

2.1 Privacy

In today's digital world, the existence of a vast number of databases that store a broad range of personal information has produced challenges surrounding privacy. Simply by linking together a subset of the available databases, it is possible to discover relevant facts and knowledge about a particular person.

The trade-off between privacy and utility of data has been the subject of much research and debate. Privacy researchers have explored various models and techniques for preserving privacy. In this section, the privacy-preserving approaches are discussed in a fine-grained manner to answer the following questions:

- What does privacy mean?
- What personal data need to be removed or obscured to achieve individual privacy?
- What available methods can be used for privacy preservation?

2.1.1 Definition of Privacy

In 1948, the Universal Declaration of Human Rights [15] acknowledged privacy as a right. Despite this, no standard definition of privacy exists [16]. Over the years, the notion of privacy has been viewed variously as referring to everything from being alone or undisturbed to our freedom from intrusion or public scrutiny to the right to anonymity [17].

The complex nature of the task of defining privacy emerges from the sheer diversity of domains that are covered by privacy [18][19]. The scope of privacy can be classified into four types: information, which is associated with the storage and management of personal data; bodily, which is concerned with physical harms caused by intrusive procedures such as drug testing; communications, which covers the privacy of any form of communication; and territorial, which is concerned with establishing boundaries for intrusion into the home as well as other areas such as the workplace or public spaces [20]. This research focuses on information privacy, which is concerned with systems that store and share data.

In the information domain, Westin [21] described privacy as "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to other people." From this standpoint, privacy amounts to an individual's right to control and manage their data. Similarly, Bertino et al. [22] defined privacy as "the right of an individual to be secure from unauthorised disclosure of information about oneself that is contained in an electronic repository." These authors emphasised the risks of privacy violation in addition to the right to control.

Based on the aforementioned views, it is reasonable to conclude that a foundational concept underpinning privacy is the right to exert control over how one's personal data is collected and handled. However, to exploit the advantages associated with recent developments in big data technology, it is necessary to collect and analyse data, which

could result in privacy breaches. For this reason, many approaches have been proposed to preserve individual privacy while handling personal data.

As shown in Figure 2-1, Stallings and Brown [23] classified privacy criteria into four main classes: anonymity, pseudonymity, unlinkability, and unobservability. Anonymity means that users cannot determine the identity of individuals bound to an operation or subject. It includes anonymity without soliciting information, which requires that the security function not ask for user identity. Anonymity needs to not conflict with authorization and access control functions which are bound to computer-based user ID not to personal user information. Pseudonymity means that users are unable to determine the identity of individual bound to an operation or subject but that individual still accountable for its actions.; pseudonymity include two sub criteria; reversible pseudonymity and alias pseudonymity. Reversible pseudonymity requires the security function to be able to determine the original user identity based on a provided alias while the alias pseudonymity requires the security function to follow specific construction rules for the alias to the user identity. Unlinkability refers to the inability to link data to a specific individual or to verify if certain specified operations were caused by the same individual. Unobservability refers to the inability to determine whether an individual's data is included in a shared dataset or whether an operation is being conducted. Unobservability include three sub-criteria; 1) allocation of information impacting unobservability which requires the security function to create a specific safeguard to prevent the concentration of privacy-related information. 2) unobservability without soliciting information which requires that the security function does not attempt to obtain any privacy-related information that could compromise unobservability. 3) requires the security function to give one or more authorised users the ability to see how resources and/or services are used. These criteria should be implemented in any trusted system to ensure user protection against discovery or misuse of identity

Gavison [24] defined privacy based on three interrelated types of privacy: secrecy, anonymity, and solitude. Secrecy refers to information that others may collect about us.

Anonymity refers to how attentive we are in public. and solitude assesses how much physical access others have to us.

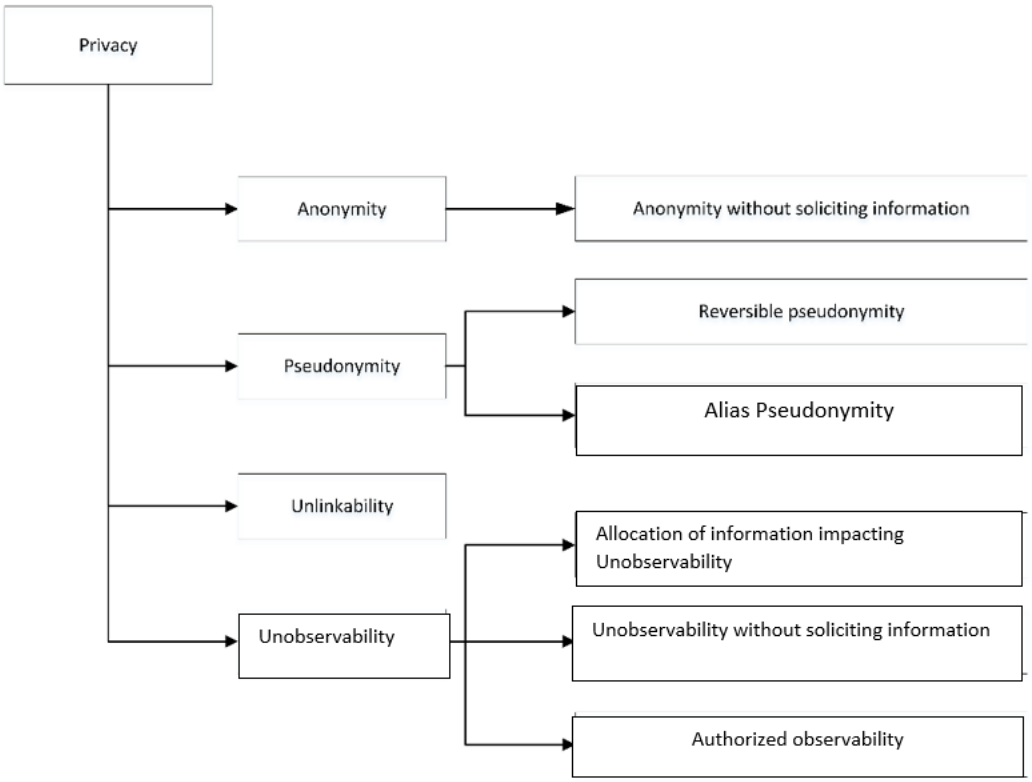


Figure 2-1: Aspects of Privacy [23]

2.1.2 Personal Data Classification

To understand the meaning and implications of the concept of identifiable information, it is necessary to focus on Article 4(1)(a) of the GDPR, in which the concept of personal data is defined as follows:

"Any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person."

From this definition, a piece of individual data can be classified based on its nature as one of four types: direct identifier, quasi-identifier, sensitive data, and insensitive attributes. A brief explanation of each type is given as follows:

- **Direct identifier:** These attributes contain pieces of information that uniquely and directly distinguish individuals (e.g., full name, driver's license, and social security number).

- **Quasi-identifier (QID):** These attributes, when combined and linked with external data, can re-identify (or reduce uncertainty about) all or some of the respondents to whom the information refers (e.g., gender, age, date of birth, and zip code).

- **Sensitive attributes:** These attributes are private and personal pieces of information that could substantially influence privacy, and which must not be linked to the data subject to which they belong (e.g., data relating to previous illnesses, medical treatments, and income level).

- **Insensitive attributes:** If disclosed, insensitive attributes will not violate user privacy. All attributes other than identifier, quasi-identifier, and sensitive attributes are classified as non-sensitive attributes. Examples of insensitive attributes may include purchase histories of goods, and mobility traces around sightseeing places

To demonstrate the importance of these attribute classifications, let us consider an example. The medical histories of a group of people are shown in Table 2-1: Example of Medical Data. The social security number attribute is a unique identifier that can be used to explicitly identify an individual. Before publishing a dataset, it is necessary to remove all direct identifiers. However, quasi-identifiers such as age, gender, and zip code also have a significant impact on identification. While quasi-identifiers do not directly identify an individual, they can provide a strong indication when linked with other publicly

available datasets. For example, a specific location (e.g., a post code) may contain only one person of a particular gender, race, and age. According to a study conducted by Sweeney [25], 87% of the population of the United States can be uniquely identified using only the tuple (Date of birth, Gender, Post Code).

Table 2-1: Example of Medical Data

ID	Age	Post code	Gender	Disease
12456	87	TW79FG	F	Ovarian Cancer
65432	65	BR 67FB	F	Ovarian Cancer
45546	48	BN54WN	M	Prostate Cancer
32658	55	SW01UB	M	Flu
76595	54	EW05HG	M	Heart Disease
87904	35	BS 97KJ	F	Heart Disease

To elaborate on the impact of quasi-identifiers, consider the sample of the voter registration roll shown in Table 2-2. Even if the ID number is removed from Table 2-1: Example of Medical Data before releasing it (as shown in the de-identified table) , it is possible to link two tables using the quasi-identifier attributes, as shown in Table 2-3. For each data tuple, this will generate a list of potential matches. In the medical release of Table 2-1: Example of Medical Data, Charli and David are the only two males on the voter registration rolls who fit a patient with heart disease. As result, it is possible to say with 50% certainty that Charli and David have heart disease. With background information about Charli and David, an attacker could increase the probability of unique identification. Likewise, Bob is the only person with the combination of age, zip code, and gender who

has prostate cancer. In this way, an individual can be uniquely identified, and sensitive information may be fully compromised by combining two sources of information. Many privacy preserving approaches have been proposed to resist this attack, known as a linking attack, by generalising quasi-identifiers or transforming the released table to satisfy the K-anonymity principle, which is discussed in more detail in the following section.

Table 2-2: Example of Voter Registration Rolls

Name	Age	Gender	Post Code
Charli R.	54	Male	EW05HG
Bob D.	48	Male	BN54WN
Carol F.	55	Male	TW80FD
Dan H.	44	Female	ED90TR
Ellen M.	65	Female	BN54WL
Anna S.	37	Female	TW898VT
Sue G.	45	Female	Bb65TR
David T.	54	Male	EW05HG
Diana G.	34	Female	SW01UB

Table 2-3: De-identified data

Age	Post code	Gender	Disease
87	TW79FG	F	Ovarian Cancer
65	BR 67FB	F	Ovarian Cancer
48	BN54WN	M	Prostate Cancer
55	SW01UB	M	Flu

54	EW05HG	M	Heart Disease
35	BS 97KJ	M	Heart Disease

2.1.3. Privacy Preserving Principle

Privacy preserving refer to the methods for avoiding the unauthenticated disclosure of information [26] include restricting access, restricting the data, and restricting the output. Restricting access by withholding and not releasing data is a relatively simple solution to the issue of privacy, but it completely eliminates the utility of available data. This is a crucial limitation because, for example, it is critical in a medical context to share useful information across research institutions. In the case of restricting the output, this involves transforming the results of user queries while leaving the data unchanged. Finally, restricting the data involves removing attributes or modifying the dataset with some form of generalisation or perturbation of values to protect the identity of the data subject. Regarding the restricted data method, this allows the sharing and distribution of data. The available techniques used to restrict the data and the output draw on various approaches to preserve data privacy, mostly for structured data. These techniques are as follows:

- **Generalisation**: This involves replacing a specific attribute value with a general value. In this operation, some values are replaced by a parent value, which is less specific in the taxonomy of attributes [27]. An example of this is representing a job attribute of a programmer or software application developer with the less specific generalised attribute of “Information Technology Specialist” [26].

- **Suppression:** This operation involves replacing the value of a piece of data with special characters (e.g., asterisks), which indicates that the values of an attribute will not be disclosed[25]. Suppression can be applied at the record

level or attribute level. An example of suppression is for the zip codes "2567459" and "2564435", which can be suppressed as "256****".

- **Data swapping**. Data swapping modifies records by switching a subset of attributes between pairs of records[26].

- **Macro-aggregation**. In macro-aggregation, individual records are never released, but aggregations of statistics over the population in the dataset are released with some level of perturbation.

- **Data redaction:** This is a data substitution technique that enables the masking (i.e., redaction) of data by removing or substituting all or part of the field value [26].

- **Randomization:** randomization include noise addition and permutation. it is a method of adding noise to the original data noise in order to make it hard to guess original data [26].

2.1.4. Re-identification Risks

In this section, re-identification risks are classified based on the risk identified by Art.29 WP [28], re-identification risks also known in the literature as disclosure risks [29], are related to the many methods by which adversaries can identify data subjects within de-identified data. Art.29 WP's Opinion on Anonymization Technique suggests [30] that data controllers should analyse the robustness of their de-identification method by considering the following three risks:

- **Singling out:** This refers to the "possibility to isolate some or all records that identify an individual in the dataset."

- **Linkability:** Defined as the "ability to link at least two records concerning the same data subject or a group of data subjects (either in the same database or in two different databases)."

- **Inference:** This denotes the "possibility to deduce, with significant probability, the value of an attribute from the values of other attributes."

Regarding re-identification through linkability, it is worth mentioning that there are three scenarios for linking de-identified records to an individual. The first scenario focuses on a specific dataset with several records relating to the same individual; in this context, an adversary identifies the data subject by linking the records together using some additional information. In the second scenario, the individual's records are contained in multiple datasets, all of which are all held by the same entity. If an adversary has access to all of the datasets within the entity (e.g., as in the case of an insider threat), they can link the records of an individual. The third scenario occurs when the individual's records are contained in multiple datasets, but these datasets are held by different entities. Based on these scenarios, it is possible to summarize the three types [31] of linkability risk:

- **Local linkability**: Denotes the possibility of linking records belonging to the same data subject within the same dataset.

- **Domain linkability:** Refers to the potential to link records related to the same data subject in two or more datasets that are in the data controller's possession.

- **Global linkability:** Defined as the potential to link records that relate to the same data subject in two or more of any datasets.

2.2 Privacy Preserving Approaches

Privacy preserving approaches are methods and tools used to release useful information while protecting data privacy. Although various typologies of privacy

preserving approaches have been proposed in the literature, this study categorised them as shown in Figure 2-2: Privacy Preserving Approaches. As indicated, there are two main categories: Privacy Preserving Data Publishing (PPDP) and Privacy Preserving Data Mining (PPDM).

2.2.1 Privacy Preserving Data Publishing

Privacy preserving data publishing, also known as non-interactive de-identification, involves publishing an entire dataset in an anonymized and non-interactive form [32]. PPDP has been studied extensively in recent years and many approaches have been proposed for different data publishing scenarios. In general, a data publishing-based privacy preserving approach may consist of the anonymization approach, where identifiers are irreversibly removed in the dataset, or the pseudonymization approach, where the identifiers are replaced with pseudonyms that can be reversed.

Anonymization Approaches

Anonymization approaches, also known as group-based approaches, are traditional techniques for privacy preservation where, to protect individual privacy, databases form groups of tuples in a way that optimises data utility and/or privacy protection, after which QID values are transformed to enforce an anonymization principle. The aim of these approaches is to irreversibly prevent the identification of the subject to whom the information belongs. This approach combines different methods to preserve data privacy

for structured data, including generalisation, suppression, and data swapping [33].

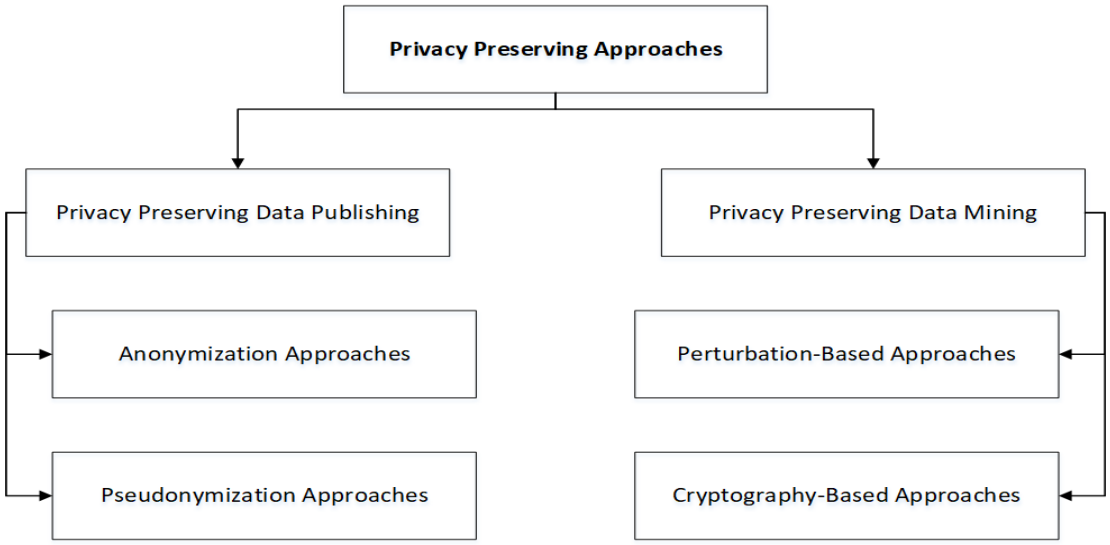


Figure 2-2: Privacy Preserving Approaches

One of the earliest group-based approaches was K-anonymity, which was introduced by Samarati and Sweeney [34]. Subsequent researchers devised different methods for data anonymization [35][36]. In the case of K-anonymity, this is defined as the property that distinguishes each record from K−1 other records based on a quasi-identifier. It means that at least K records are required in each equivalence class to achieve anonymity as shown in Figure 2-3: K-Anonymity [39]. For example, if all records in a table satisfy the requirement of K-anonymity, then for some value of K, a record can be identified with 1/K confidence if quasi-identifiers are known. K-anonymity focuses on quasi-identifier attributes and invests no effort in sensitive attributes. Consequently, it is susceptible to multiple attacks, including homogeneity attacks [37] and background attacks [38].

K = 2

	Race	Birth	Gender	ZIP	Problem
t1	Black	1965	m	0214*	short breath
t2	Black	1965	m	0214*	chest pain
t3	Black	1965	f	0213*	hypertension
t4	Black	1965	f	0213*	hypertension
t5	Black	1964	f	0213*	obesity
t6	Black	1964	f	0213*	chest pain
t7	White	1964	m	0213*	chest pain
t8	White	1964	m	0213*	obesity
t9	White	1964	m	0213*	short breath
t10	White	1967	m	0213*	chest pain
t11	White	1967	m	0213*	chest pain

Figure 2-3: K-Anonymity [39]

To overcome the limitations of K-anonymity, different algorithms, models, and frameworks have been proposed. For example, the l-diversity privacy principle requires that a sensitive attribute should include at least l well-represented values in each group of the K-anonymized data to solve the issue of the homogeneity attack within K-anonymization. To address the background knowledge problem associated with the l-diversity technique, the idea of t-closeness was proposed [36]. It requires that attributes in an equivalence class and table have a close distribution that does not exceed a threshold t. The l-diversity, t-closeness, and other privacy technical models, which focus on the protection of sensitive values in structured data, typically assume that there is one (or a few) predefined sensitive attribute, which is not a realistic scenario for text data. Clinical text documents usually consist of a group of unstructured (not predefined) sensitive attributes, such as symptoms, conditions, test results, diagnosis, and treatments. Applying the concept of l-diversity or t-closeness to sensitive attributes in data from medical documents is difficult, if not impossible [40] [41] [42]. Furthermore, group-based de-identifications are difficult to apply for structured big data because the reduction of

information loss depends on the repeated scanning of the data during the anonymization procedure, which may be complex for big data due to its characteristics [43].

Although the terms “anonymization” and “de-identification” are often used interchangeably, there is a significant difference between them. Kushida et al. [44] stated that the de-identification of data refers to the process of removing or obscuring any personally identifiable information from individual records in a way that minimises the risk of disclosure of the data subject’s identity. However, de-identified datasets can contain encrypted identifiers where only authorised individuals have access to the encryption key. The existence of a key makes it possible to recover the original data for users with correct authorisation. Anonymization, on the other hand, refers to a data de-identification process that produces de-identified data that cannot be reversed back to the original data.

Pseudonymization Approaches

Pseudonymization is a particular type of de-identification in which information that directly identifies an individual such as (names, phone numbers, Social Security Number) are replaced with pseudonyms. Pseudonymization enables linking personal data through various dataset, when all identifiable information is consistently pseudonymized. pseudonymization process can be reversible or irreversible. In reversible pseudonymization data can be recovered when the link between original identities and the pseudonyms is maintained or if the replacement done with an algorithm whose parameters are known. It provides an option the de-identification process to be reversed at some time in the future and re-identifying the data subjects. For example, identifiable information can be encrypted with a secret key to create a pseudonym; decrypting the key reversed the pseudonymization process, recovering the original identifier. In irreversible, the pseudonymized data do not contain information that permits the link between the pseudonymized data and the data subject to be re-established such as using one-way

function such as cryptographic hashing. Irreversible pseudonymization might overlap with anonymization but it keeps the pseudonym consistent all across the resulting data set. However, a well-known attack against it is the “rainbow attack”, where an adversary recovers the plaintext using precomputed tables [45].

When the GDPR came into effect in 2018, it introduced the pseudonymization as privacy preserving concept that is more restrictive than merely removing or masking direct identifiers. GDPR define pseudonymization as

“The processing of personal data in such a way that the data can no longer be attributed to a specific data subject without the use of additional information, as long as such additional information is kept separately and subject to technical and organisational measures to ensure non-attribution to an identified or identifiable person” [46]

In order to meet GDPR standard for pseudonymization, data controllers should implement several technical and organizational measures to ensure that pseudonymous data is disconnected from the key enabling re-identification.

Pseudonymization is encouraged and rewarded throughout the GDPR (will discussed in more details over next section. For example, it is suggested by GDPR as protective measure for processing data (e.g., for research or analysis) other than the data subject consent. However, GDPR restricts a data handler's potential to benefit from pseudonymized data if re-identification processes are "reasonably likely to be employed, such as singling out, either by the controller or by another person to identify the natural person directly or indirectly" [47] As a result, whether or not pseudonymization will result in exemption from these GDPR obligations will be determined by the strength of the method and implementation used.

2.2.2. Privacy Preserving Data Mining

Privacy preserving data mining refers to a group of methods that enable the extraction of relevant knowledge from a large amount of data, all the while preventing sensitive data or information from disclosure. In PPDM[45], data are not shared but are used instead for statistical processing or machine learning. The calculated results may be released in the form of statistical tables based on summarisation and aggregation, as well as classifiers that implement machine learning algorithms. PPDM methods can be broadly classified into perturbation-based and cryptographic-based approaches.

Perturbation-based Approaches

Perturbation-based approaches are characterised by the use of the e-differential privacy techniques originally introduced by Dwork [48]. The privacy guarantee provided by Differential Privacy (DP) is implemented via the addition of noise to the output of the computation, thereby meaning that curious users cannot infer whether the individual data was involved in the computation. Differential privacy sanitises the dataset in a way that the query result cannot be used to infer much about a specific individual. It is defined as follows:

A randomised function *K* satisfies e-differential privacy if for any two datasets D_1 and D_2, which differ in only one row for all $S \subseteq$ range *K*, the following inequality holds:

$$\frac{\Pr[k(D_1) \subseteq S\,]}{\Pr[k(D_2)\ \subseteq S\,]} \leq \varepsilon \tag{2.1}$$

Where the S represents all potential output of K that could be predicted.It specifies the amount of privacy protection. And parameter ε bounds the ratio of the probability distributions of the two datasets D_1 and D_2. where the smaller value of ε the more difficult it will be for an attacker to determine an individual's data. Hence, this leads to better protection.

Differential privacy provides a mechanism for obtaining useful information from databases containing personal information without revealing the personal identities of the individuals. This is achieved by adding noise to the query result using the Laplace mechanism [48]. Analysts in DP are not given direct access to a database containing personal information. Instead, an intermediary piece of software is introduced between the database and the analyst called privacy guard to protect the privacy [49] . Figure 2-4 illustrates the process of differential privacy as follow:

1- The analyst can request the database via this intermediary privacy guard.
2- The privacy guard evaluate this request and other earlier request in term of privacy risk
3- The privacy guard then obtained the result from the database.
4- Add some noise to the result based on the evaluated privacy risk and finally provide it to the analyst.

The added noise is sufficiently large to protect the privacy and at the same time it is sufficiently small to ensure that the information provided to the analyst is still useful.

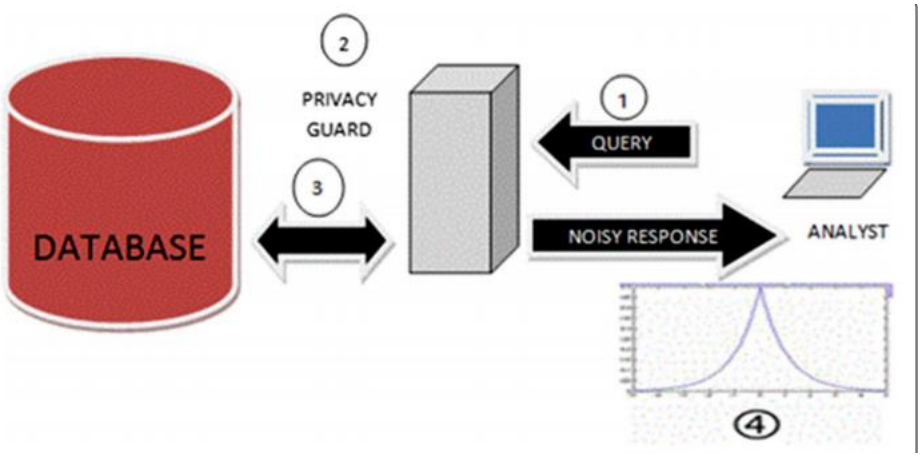


Figure 2-4: Laplace Mechanism [49]

Cryptographic-based Approaches

Cryptographic-based PPDM approaches, also known as secure multiparty computation (SMC) methods, allow the construction of a data mining model from

distributed data using cryptographic approaches that protect the discovery of individual data items between the parties. [50] . The key component of these methods is the algorithm, which uses cryptographic computations to duplicate a distributed data mining algorithm without disclosing data from any site. It aims to produce the same results as those mined from the original dataset.

The basic task of SMC is to compute functions over inputs provided by multiple recipients without actually sharing the inputs with one another. To give a common example, consider a situation where two individuals are interested in knowing who is wealthier without revealing their net worth to one another. Abstractly, this problem simply involves comparing two numbers, each held by one party, without either party revealing its number to the other.

A well-known SMC technique is homomorphic encryption, which allows a set of calculations to be conducted on ciphertext and subsequently generates an encrypted result that matches the result of calculations to be conducted on the plaintext when decrypted. Homomorphic encryption can be categorised as either partial homomorphic encryption or fully homomorphic encryption. To compute specific purpose functions, partial homomorphic encryption executes one sort of operation (e.g., addition or multiplication) [51][52]. Fully homomorphic encryption was proposed by Rivest et al. [53] as a way to execute a set of homomorphic operations, such as addition and multiplication. Homomorphic encryption involves computationally expensive public-key procedures, which scale inefficiently with increasing security parameters [54]. As discussed in [55], while this technique provides strong privacy guarantees, it does not scale effectively for large amounts of data due to the usage of heavyweight cryptographic operations between parties.

To summarise, privacy is preserved in privacy-preserving data publishing by employing privacy techniques that alter the original table to prevent information

disclosure. In contrast, in privacy-preserving data mining, privacy is maintained by using privacy techniques that alter the result of the queries or the statistical tables. Each model has both advantages and disadvantages, as summarised in Table 2-4.

Table 2-4: Comparison of privacy-preserving approaches (anonymization based) in terms of the employed privacy preserving principle

Privacy Model	Privacy Preserving Principle	Description	Advantages & Disadvantages	Applications and Domains
K-anonymity	Generalization, Suppression	Anonymity is guaranteed by the existence of at least other k-1 undistinguishable (with the QID) records for each record in a database. This group of k undistinguishable records is referred to as equivalence class.	[+] simplicity of definition [+] great use of existing algorithms [0-] Assumes each record is unique. [-] sensitivity attributes are not taken into consideration for anonymization	Wireless Sensor Network[56]; Location based services[57], Cloud[58], E-health[28], Public Transport [59]
l-diversity	Generalization, Suppression	Expands the k-anonymity model by requiring every equivalence class to have at least 1 "well-represented" value for the sensitive attributes	[+] Consider diversity of attributes [-] does not consider distribution of sensitive value that can cause privacy breach in skew distribution.	E-health [28], Location-based service [60]
t-closeness	Generalization, Suppression	Solves the l-diversity problem of skewed sensitive values distribution by requiring that distribution of the sensitive values in each equivalence class to be "close: to the corresponding distribution in the original table, where close means upper bounded by a threshold t.	[+] takes into consideration the distribution of sensitive values for equivalence class. [-] the information about the correlation between QID attributes and sensitive attributes is lost as t decreases (as privacy increases)	Location-based service [61]

Table 2-5: Comparison of privacy-preserving approches in terms of the employed Privacy Preserving Principle

Privacy Model	Privacy Preserving Principle	Description	Advantages & Disadvantages	Applications and Domains
Pseudonymization	Data redaction	Achieved by removing the association between identifiable data and the data subject and introduces a new identifier that establishes a bidirectional-mapping between that subject and the new identifier	[+] original data can be recovered by authenticated user. [+] easy to implement specifically for structured data. [-] weak pseudonym algorithm such as hashing is vulnerable to rainbow attack. [-] static pseudonymous is vulnerable to linking attack.	Healthcare [62],Clinical trial [12], Location-based service [63].
e-differential privacy	Randomization	Ensures that a single record does not considerably affect the outcome of the analysis of the dataset	[+] provides a formal privacy guarantee and a solid privacy loss metric [+] ensures that the participation of a single individual does not result to a privacy violation greater than the obtained from the non-participation of the same individual. [-] No guide to establish e as it is dependent on the dataset. [-] privacy guarantees can involve heavy data perturbation for numerical data.	E-health [64], Smart meters [65], Spatial crowdsourcing [66], Wireless Sensor Network [67]
Secure Multiparty Computation Methods	Homomorphic encryption		[+] strong privacy guarantee [-] have a large computational overhead	Cloud work load protection[68]

As mentioned earlier, most of the existing privacy-preserving techniques have been developed long before GDPR requirements were established and filled to comply with several of its provisions. In the next section, GDPR requirements for preserving individual privacy will be discussed in more detail.

2.3.Data De-identification in GDPR

The new GDPR came into effect in 2018 and replaced the old EU data protection legislation, the Data Protection Directive [69]. In the 1995 Directive, data is either personal data (and, therefore, is subject to data protection regulations) or it is anonymous (and, therefore, is not subject to data protection regulations). Unlike the 1995 Directive, the GDPR recognises an intermediate level of de-identification by explicitly introducing the concept of pseudonymous data. Furthermore, Article 11 [70] of the GDPR describes another level of de-identified data. This section first discusses the GDPR principle for de-identification and then outlines different levels of data de-identification within the GDPR.

2.3.1. GDPR Principles for Processing Personal Data

The GDPR sets out seven key principles relating to the processing of personal data: lawfulness, fairness, and transparency; purpose limitation; data minimisation; accuracy; storage limitation; integrity; and confidentiality and accountability.

This section discusses the three principles that are related to privacy preservation through de-identification, namely:

- **Data minimisation:** Data processing should be “adequate, relevant, and limited to what is necessary in relation to the purposes for which they are processed.”

- **Storage limitation:** This limitation means that an individual's personal data can be retained for no longer than is necessary to carry out the purpose for which the data is processed, as mentioned in principles relating to

processing of personal data[6] [1] of the GDPR. In particular, data can be "kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed; personal data may be stored for longer periods insofar as the personal data will be processed solely for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes in accordance with Article 89(1)" [2]

- **Purpose limitation:** This limitation restricts the use of personal data beyond the purpose for which the data was originally collected, as mentioned in principles relating to processing of personal data[7] [1] of the GDPR. In particular, data can be "collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1)" [2]

Article 89(1) specifies the circumstances in which the processing of personal data is permitted, as well as the safeguards that must be in place, as follows:

"Processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes, shall be subject to appropriate safeguards, in accordance with this Regulation, for the rights and freedoms of the data subject. Those safeguards shall ensure that technical and organisational measures are in place in particular in order to ensure respect for the principle of data minimisation. Those measures may include pseudonymization provided that those purposes can be fulfilled in that manner. Where those purposes can be fulfilled by further processing which does not permit or no

[6] GDPR Article 5(e)
[7] GDPR Article 5(b)

longer permits the identification of data subjects, those purposes shall be fulfilled in that manner."

The analysis of these new data protection principles indicates that GDPR principles do not prevent the use of personal data for analytics or other useful secondary applications. However, they require the implementation of new technical and organisational measures to protect that data. The GDPR explicitly points to data de-identification as a measure that can support meeting the requirements of several of its provisions.

2.3.2. The Three Types of Data in GDPR

This section introduces the three types of de-identified data that emerge from GDPR's principles analysis, and it interprets their underlying meanings.

Anonymized data: Refers to data that do not relate to "an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable" [47] . For example, the data can be aggregated to a larger level or transformed into statistics such that the individuals whom the data describe remain anonymous.

To assess the possibility of identification, all reasonable means for converting the data back to an identifiable form must be considered. Such singling out can identify the nature of a person either directly or indirectly. It is necessary to take into account factors such as the cost and time required for identification, as well as the available technologies. Anonymized data do not consider personal data, and so are not subject to data protection regulations.

Pseudonymized data: Refers to data that have been processed in such a way that "the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organisational measures to ensure that the personal data are not

attributed to an identified or identifiable natural person" Article 4 [46]. In other words, Pseudonymized data are de-identified data that relate to an individual but do not reveal the identity of that individual; these data do not include any identifiers, and they cannot be directly linked to the data subject. However, there is a known way (algorithm, pseudonym, or key) to allow the data controller to link or re-link Pseudonymized data with identifying data.

To understand the GDPR definition, it is important to distinguish between the two terms "identified" and "identifiable." A person is generally considered to be identified if the data can be linked to their true identity. On the other hand, the term "identifiable" refers to the ability to identify a person who has not been identified but rather is characterised in the data in such a way that they can be discovered if research is carried out using additional information (e.g., a key) or background information. This explains why Pseudonymized data – at least potentially – is still regarded as personal data and, as such, is subject to data protection regulations. However, pseudonymization is highly encouraged across the GDPR. For example, under Article 89(1) [2] , pseudonymization is recommended as an "appropriate safeguard" that must be in place for retaining or processing data for archiving purposes in the public interest, for scientific or historical research purposes, or for statistical purposes.

Article 11 data: Refers to de-identified data that meet the standard set out in Article 11, which relates to whether the "controller is able to demonstrate that it is not in a position to identify the data subject." First, Article 11(1) [70] states, "If the purposes for which a controller processes personal data do not or do no longer require the identification of a data subject by the controller, the controller shall not be obliged to maintain, acquire or process additional information in order to identify the data subject for the sole purpose of complying with this Regulation."

Additionally, Article 11(1) [70] specifies that in such cases, if the "controller is able to demonstrate that it is not in a position to identify the data subject." Article 11 data are de-identified data that may include pseudonymized data, which are data related to an individual but that do not reveal the identity of the individual, do not include any identifiers, and cannot be directly linked to the data subject. In contrast with pseudonymized data, there is no known way for the data controller to reverse the pseudonymization process. An example is when one data controller pseudonymises data and transfers it to another data controller who does not have access to the additional information. The GDPR specifies that de-identified data that meet the Article 11 standard do not need to comply with specific rights of the data subject, including the right of access [71] , rectification [72], erasure [73], processing restrictions [74], data portability [75] .

The three levels of de-identified data used in this chapter and described above summarised in Table 2-6 as follows:

Table 2-6: Summary of the De-identified Data under GDPR

	Anonymized data	**Pseudonymized data**	**Article 11**
Directly linked to identifying data	No	No	No
Known way to re-identify	No	Yes	No
Considered personal data	No	Yes	Yes

2.3.3. Effectiveness of Data Sanitization Techniques to Mitigate Re-identification Risk

Article 29 WP [30] classifies data sanitization approaches into randomization, generalization, and masking direct identifiers. k-anonymization, L-diversity, and t-closeness are included in generalization. Noise addition, permutation, and differential privacy are included in randomization, while pseudonymization is included under the

masking method. This section discusses the robustness of the sanitization methods in relation to singling out, inference, local, domain, and global linkability risks.

- **k-anonymization**: In k-anonymization, an individual record is distinguished from $k-1$ other records based on a quasi-identifier. This method mitigates singling out and local linkability risks because the possibility of connecting two records to the same data subject is less than 1/k. Significantly, k-anonymization cannot eliminate domain and global linkability risks, as shown in Section (2.1.2), given that it is possible to link records from an anonymized medical dataset with records from a voter registration dataset using an intersection attack.

- **L-diversity**: The key enhancement of l-diversity over k-anonymity is that it guarantees that the sensitive attribute in each equivalence class (i.e., the k group) has a minimum of l different values. As a result, it eliminates the risk of inference to a probability of no more than 1/l. However, it is similar to k-anonymity in that it cannot mitigate domain and global linkability risks; this is because records can still be linked together if they have the same sensitive attribute values.

- **T-closeness**: This sanitization procedure requires that attributes in an equivalence class and table have a close distribution that does not exceed a threshold t. Similar to the previous techniques, t-closeness can mitigate singling out and inference attacks, but global linkability is still an issue.

- **Randomization**: This technique involves adding noise or shuffling the values of attributes within the dataset, making the values of such attributes less accurate or imprecise. The method, however, cannot mitigate risks associated with local, domain, and global linkability. Indeed, this method

only reduces the reliability of linking records to the data subject as the attribute value is inaccurate.

- **Differential privacy**: This implies that based on the query result, it is impossible to determine whether a data subject is included in a dataset. When numerous inquiries on one or more datasets are permitted, the queries must be monitored and the noise appropriately adjusted to guarantee that adversaries cannot infer more information based on the results of multiple queries. Hence, "may not" is assigned to the risks based on whether queries are monitored.

- **Pseudonymization (static pseudonym)**: This technique uses cryptographic techniques such as hashing to mask identifiable information to reduce the linkability between the records and data subjects. However, if only direct identifiers are masked, it is possible to single out an individual within the dataset using quasi-identifier attributes. It is also possible to link the data subjects with records in one or more datasets if the same pseudonym is used for the same record or attribute. Hence, pseudonymization using static pseudonyms is susceptible to domain and global linkability risks.

- **Pseudonymization (dynamic)**: This technique replaces identifiable information with dynamic pseudonyms that change for the same attributes, thereby disassociating the attributes from the data subject and preventing susceptibility to singling out risk. Furthermore, local and global linkability risks might be eliminated if the direct and indirect identifiers are replaced with dynamic pseudonyms, as well as if organizational and security measures are implemented, thereby ensuring that pseudonymous data are disconnected from the key that enables re-identification. Therefore, "may

not" is assigned to the risks depending on whether security measures are implemented for the re-identification key.

Table 2-7 summarizes the risk assessment for each data sanitization technique in terms of re-identification risks.

Table 2-7: Robustness of data sanitisation methods against risks

	Singling out risk	Local linkability risk	Domain/global linkability risk	Inference risk
K-anonymity	No	No	Yes	Yes
I-diversity	No	No	Yes	No
t-closet	No	No	Yes	No
Randomization	Yes	Yes	Yes	Yes
Differential privacy	May not	May not	May not	May not
Pseudonymization(static)	Yes	Yes	Yes	Yes
Pseudonymization(dynamic)	May not	May not	May not	May not

2.3.4 Obligations Under the GDPR

The GDPR imposes a variety of obligations. This section examines six key GDPR obligations to illuminate how pseudonymization and anonymization influence their applicability.

1. Provide Notice to the Data Subject (Right to be Informed*)*

Transparency is a fundamental principle of data protection rules, along with the obligation to inform data subjects about the collection, use, and disclosure of personal information. The GDPR requires the data controller to provide comprehensive details to the data subject about their purposes for processing data, the retention period, and

who data will be shared with [76] [77] [71]. In its language, the GDPR does not distinguish between fully identified personal data and pseudonymized personal data. Consequently, this obligation applies to both pseudonymized data and Article 11 data. However, the obligation does not apply to anonymized data as it is not considered personal data.

2. Consent and the Legal Basis for Processing

The GDPR requires the existence of a legal basis for the processing of personal data, such as explicit consent from the data subject [78]. However, obtaining explicit consent can be challenging and impractical in some cases, with examples including big data mining and machine learning research.

Article 89(1) of the GDPR identifies pseudonymization as one of the protective measures that controllers can use to evaluate the feasibility of further processing of personal information for a secondary use, specifically where "processing for another purpose is compatible with the purpose for which the personal data are initially collected" [2]

Therefore, in some circumstances, pseudonymized data, including Article 11 data, can be processed on a basis other than the consent of the data subject. Fully anonymized data are outside the scope of the GDPR and, hence, they are free of these obligations.

3. Data Subjects' Rights

The GDPR establishes a new right for data subjects, enabling them to request the deletion of personal data. This is known as the "right to erasure" or the "right to be forgotten." [73] If specific conditions are met, data controllers must react to such requests and delete personal data "without undue delay." Implementing such a right can be difficult, especially if many copies need to be located, mapped, and deleted.

For example, in a clinical trial that maintains data only about each patient's ID, gender, and age, it would be difficult to find and map a specific record to delete it.

Additional rights given to the data subject under the GDPR include the rights of access [71], rectification [72], and data portability [75], along with the right to object to the processing of personal data [74]. Similar to the right to be forgotten, complying with these rights can be challenging. However, data that is de-identified based on Article 11 [70], where the "controller is able to demonstrate that it is not in a position to identify the data subject," is exempt from this obligation; in such cases, it is not necessary to comply with the data subject rights such as request of data deletion or processing restrictions.

4. Data protection by default and by design

The GDPR introduced a new requirement known as data protection by design and by default. This requires the data controller to "implement technical and organizational measures such as pseudonymization, which are designed to implement data-protection principles, such as data minimisation" [79]. The GDPR further states that this should be undertaken at the earliest stages of the design of the processing operations. These principles are required for both identified and pseudonymized data. However, since pseudonymization is mentioned as an organizational measure to satisfy or to meet the data minimisation requirement, it will at least partially fulfil the criterion of data protection by default.

5. Data Breach Notification

In the event of personal data breaches, the GDPR adds new rules for informing supervisory authorities and/or data subjects. The data controller must alert supervisory authorities to the personal data breach unless it is "unlikely to result in a risk to the rights and freedoms of natural persons" [80]. It is also necessary for the data controller

to inform the data subject, if "the personal data breach is likely to result in a high risk to the rights and freedoms of natural persons" [81] .

The risk assessment for this obligation depends on the level of de-identification. For example, fully identified personal data will definitely lead to high risk if it is breached, and so a notification is required in this case. The GDPR acknowledges that the use of pseudonymization can "can reduce risks to the data subjects concerned and help controllers and processors meet their data-protection obligations." [82]. Thus, pseudonymization can help data controllers to mitigate their breach notification obligations. In other words, while the breach notification is required in the case of identified data, it is less likely to be required in pseudonymized data, and for fully anonymized data, it is not required at all.

6. Data Retention Limitations

As mentioned in Section 2.3.1 , the imposition of storage limitations is an essential principle under the GDPR. In particular, data controllers are required to retain data no longer than is necessary to carry out the purpose for which the data were collected. However, in some circumstances, data can be retained for a longer period [1] such as for conducting scientific or statistical research, and also when appropriate safeguard measures are implemented (e.g., pseudonymization). Once again, fully anonymized data are not considered personal data and can be retained indefinitely.

Table 2-8 below highlights the advantages of each type of data and illustrates whether or not the different obligations under the GDPR apply. To Sum up, pseudonymisation serves as vehicle to relax certain data controller obligation including: lawful repurposing 'further processing' in compliance with GDPR limitation principle, Archiving data for statistical or scientific research processing and for reduce notification obligation in case of data breach. It also can be used as technical protective measure to reduce the risk within the meaning of privacy by design.

Table 2-8: Summary of different types of data and obligations required

Different Obligations under the GDPR	Anonymized data	Article 11	Pseudonymized data
Provide notice to data subject	Not required	Required	Required
Consent and the legal basis for processing	Not required	Potentially helps	Potentially helps
Some data subject rights	Not required	Not required	Required
Data protection by design	Not required	Met	Met
Data breach notification	Not required	Less likely required	Less likely required
Data retention limitation	Not required	Potentially helps	Potentially helps

The above discussion and summary table clarify that the new rules and privacy principles introduced by the GDPR do not restrict the use of personal data for analytics or other useful secondary applications. However, they do necessitate the implementation of new technical and organizational safeguards to preserve the privacy of data subjects. One of these measures is strong pseudonymization, which can involve encrypting identifiable information to reduce the risk of identifying the data subject. In the next section, an examination is given of well-known encryption approaches, along with an evaluation of their feasibility regarding their use in data de-identification.

2.4. Cryptography

Cryptography is a research field in mathematics and computer science that is concerned with the use of techniques to secure the transfer of messages between two parties. Cryptography consists of a set of processes or functions that involve the use of keys to encrypt plain text, ensuring that only the intended recipient of a message can read and process it. Cryptographic algorithms can be divided to classic and modern cipher. Classic ciphers are character-oriented ciphers which can only be used to encrypt text. Modern ciphers on the other hand are bit-oriented ciphers that can be used to encrypt any form of data. More detail about each type can be found in Appendix V .The most popular examples of modern ciphers are Advanced Encryption Standard (AES) and Data Encryption Standard (DES). AES uses permutation-substitution structure , which involves a series of substitution and permutation processes to produce the encrypted block. DES uses the Feistel network structure which divides the block into two halves before starting the encryption steps [83]. Most of the existing encryption algorithms are driven by these structures. However, the cumbersome key management and distribution of these approaches do not provide a suitable level of scalability specifically in big data emerging application [84] . Therefore, more lightweight and practical alternatives need to be developed [14].

More detail about the encryption structure for AES and DES can be found in Appendix V

2.4.1 Cryptography Evaluation and Performance Criteria

Encryption algorithms are usually measured and evaluated based on the criteria of security and performance, the latter of which is easier to measure. Typically, performance is measured by computing the encryption execution time, which can be achieved by calculating the time taken by an encryption algorithm to processes a file during encryption and decryption. In addition to speed, encryption algorithms also are evaluated according to resource requirements (e.g., memory consumption during the encryption process and the size of the generated ciphertext).

Security, on the other hand, is a comparatively difficult criterion to assess in relation to encryption algorithms because there is no absolute judgment about whether a certain algorithm is secure. Usually, the designers of encryption algorithms perform security analyses when they publish their work, and they generally claim that the design is secure until successful cryptanalysis is published demonstrating that the algorithm is no longer secure. Cryptanalysis of this kind often aims to identify weaknesses in the algorithm, where the intention is to exploit the part/round where the cipher fails to satisfy the diffusion or confusion properties discussed at the beginning of this chapter. In general, the security analysis of encryption algorithms is performed by applying statistical and analytical tests that measure the levels at which confusion and diffusion are satisfied. The next section describes and explains several of these statistical and analytical tests.

- ***The Avalanche Effect***

The avalanche effect is a desirable feature of any cryptographic algorithm, which attempts to reflect the idea of high nonlinearity [85]. If a significant level is not demonstrated during an avalanche test, then the algorithm's randomisation is inadequate;

this can allow a cryptanalyst to make predictions about the plain-text only from the given cipher-text. For an algorithm to satisfy the avalanche criterion, a slight change in the input (flipping a single bit in either the plain-text or the key) produces a significant change in the output (at least half of the bits are flipped) [86]. When the avalanche property is satisfied for an algorithm then it is said to exhibit a good diffusion.

- ***Bit Independence Criterion (BIC)***

The BIC states that each pair of bits in the cipher-text for a cryptographic algorithm should be bit-independent. Webster and Tavares [87] defined the BIC for a cryptographic algorithm as follows: a function satisfies the BIC if any input bit i is inverted in the plain-text or the key, then the output bits j and k in the cipher-text must change independently. If the BIC is satisfied, then it is challenging to infer one value of the sequence from the others. The absolute correlation coefficient can be used to describe the bit independence (BI) between bits j and k in the cipher-text as follows:

$$BI\left(C(x_j), C(x_k)\right) = corr\left(x_j^1 \dots \dots \dots\ x_j^i \dots \dots x_j^N\right), \left(x_k^1 \dots \dots \dots\ x_k^i \dots \dots x_k^N\right) \qquad (2.2)$$

where $C(x_j)$ and $C(x_k)$ denote the j^{th} and the k^{th} bits in the cipher-text, and x_j^i and x_k^i denote the value of j^{th} and k^{th} bits in the cipher-text when i^{th} changes in the plain-text. To judge the results, according to Cohen [88], the value of the outcome is summarised as follows:

1. If the result is 0.5, then the relationship between $C(x_j)$ and $C(x_k)$ is strong.
2. If the result is 0.3, then the relationship between $C(x_j)$ and $C(x_k)$ is moderate.
3. If the result is 0.1, then the relationship between $C(x_j)$ and $C(x_k)$ is weak.

- ***Frequency Analysis***

In cryptanalysis, frequency analysis refers to the study of the frequency of characters in a cipher-text. Frequency can be visualized using histograms [89], where the distribution of the characters in the document is graphically represented. In other word certain letters and combinations of letters occur with varying frequencies. For example, vowel A, E, I, O, and U are the most frequently occurring characters in the text while Z, Q and X are rarely occurring characters. Likewise, TH, ER, ON, and AN are the most common pairs of letters in the text. Figure 2-5: Letter and their Relative Frequency [90] shows English letter and their relative frequency. An adversary can use frequency analysis to discern the key or plain-text; this type of attack is called a statistical attack. To prevent such attacks, the distribution of each character should differ in both the plain-text and cipher-text. Also, the histograms for the plain-text and cipher-text should not be statistically similar. Furthermore, the histogram for the cipher-text should be relatively uniform, meaning that it will not supply any useful information related to the plain-text under the use of cryptanalysis. Then the differentiate cryptanalysis be more difficult.

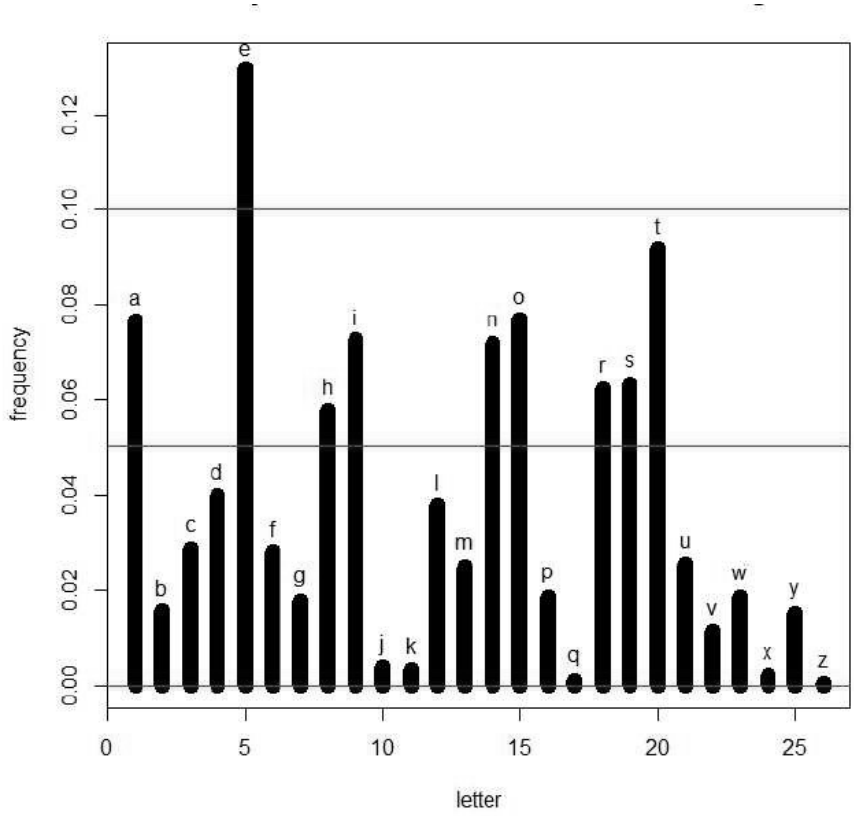


Figure 2-5: Letter and their Relative Frequency [90]

In addition to the statistical and analytical tests mentioned above, the encryption algorithm's strength can be measured by its robustness against attacks. In the next section, different attacks are discussed.

2.4.2. Security against Attacks

A cryptographic attack is a means of breaking a cryptographic system's security by identifying a flaw in a code, cipher, or key management mechanism. This procedure is also known as cryptanalysis. A description of some known attacks is given in the following section.

Brute Force Attack

A brute force attack, also known as an exhaustive search, is an attack in which every potential key is tried in an attempt to decrypt the code. Theoretically, brute force attacks can work against almost every symmetric encryption algorithm, but in practice,

these attacks are generally ineffective. This is because as the key size increases, brute force attacks become time-consuming. For example, if an adversary knows that the key size involved in a cryptosystem is 10 bits long, they can examine 2^{10} different keys, which is feasible. By contrast, if the key length is 128 bits, it is necessary to try 2^{128} potential keys, which is widely considered out of reach even for today's most advanced technology. In the latter case, even a computer capable of executing billions of operations per second would require billions of years to examine every key [91].

Differential Cryptanalysis

Differential cryptanalysis was introduced in 1990 by Biham and Shamir [92]. It is a chosen-plaintext attack in which the adversary selects plaintexts and then performs encryption with the same key for each, where the key is unknown to the adversary. The plaintexts are in pairs in this context, and each plaintext has a specified difference relative to its paired plaintext.

Consider the two plaintexts $P, \check{P} = P \oplus X_p$ with their corresponding ciphertexts $Enc_k(P) = C,\ Enc_k(\check{P}) = \check{C}$. Let $X_{C=}\ C \oplus \check{C}$. If there exists a value X_C that occurs with a higher probability than would be if C and P were selected independently at random and not related through the encryption algorithm, then the cipher function can be differentiated from a random function. In this scenario, the differential relation is often used to retrieve part of the secret key.

$$\Pr\left[\ Enc_k(P)\ \oplus Enc_k\big(\ P\ \oplus\ X_p\ \big) = X_C\ \right] = \frac{1}{2} + \ \beta\ where\ \beta > \frac{1}{2} \qquad (2.3)$$

Meet-in-the-Middle Attack

The meet-in-the-middle attack was developed in 1977 by Diffie and Hellman, as explained in [93]. It is a type of known-plaintext attack in which the adversary knows some part of the plaintext and its corresponding ciphertext. The adversary tries to go from both ends (i.e., plaintext and ciphertext) to an intermediate state. Thus, the

adversary encrypts the plaintext $Enc(P, k1)$ for a number of iterations, after which the ciphertext $Dec(C, k2)$ is decrypted for some iterations, to attain the same middle state of the cipher. This can be written as follows:

$$Enc(P, k1) = Dec(C, k2) \quad (2.4)$$

Where C is the ciphertext corresponding to the plaintext P, both of which are known to the adversary.

The initial stage in the attack is to generate a table that contains all potential values for one side of eq. (2.4). In turn, the values are computed for the opposite side of eq. (2.4), after which they are compared to the values for the first side of the equation and saved in the table. The adversary looks for a pair of secret keys, $k1$ and $k2$, for which the value of $Enc(P, k1)$ in the table equals the computed value of $Dec(C, k2)$.

2.5. Dynamic Key

Dynamic keys are one-time symmetric cryptographic keys that form a series of keys [75]. Every plaintext in the system is encrypted using a different cryptographic key, much like a one-time pad. As a result, any effort to compromise the cryptographic system by reusing a compromised key will be identified easily. The dynamic keys are created offline by participating parties rather than being distributed across the parties. Unlike session keys, which are transferred between parties during each session, there is no key exchange at every session or transaction. A dynamic key generation system is used to generate a series of dynamic keys from initial parameters. The initial parameters can be exchanged only once at the start of the session using the key exchange protocol. A mathematical representation of a series of dynamic keys is as follows:

$$n \in N, n > 1, \{DK_i\} = \{DK_1, DK_2, \ldots\ldots, DK_n\} \quad (2.5)$$

Where n is the number of dynamic keys in a sequence.

The sequence of dynamic keys must be as secure as possible against cryptanalysis attempts. In other words, recovering one or more cryptographic keys in the sequence should not recover the entire series and, in this way, undermine the whole system's security.

The condition that underlies the dynamic generation scheme can be explained mathematically as follows: for every algorithm A, the highest probability that A can correctly guess the current dynamic key DK_m from the previous dynamic keys DK_i is S, where s denotes the bit length of DK_m.

$$\forall\, i,m \;\in N \;\; 1 < i \; < m, \;\; P(A\{DK_i\}) = DK_m \;\leq \frac{1}{2^s} \qquad (2.6)$$

The produced sequence must have the following characteristics:

1. The dynamic key sequence must be unique to both the sender and the recipient. To meet this criterion, the scheme must generate the same series of dynamic keys using the same initial parameters.

2. Compromising prior dynamic keys must not expose current and future dynamic keys in the sequence to vulnerability. An adversary must not be able to compute any other dynamic keys DK_{n+1} from one or more previous compromised keys $DK_i, 1 \leq i \leq n$. In other word, compromising previous dynamic keys must not create vulnerability for current or future dynamic kyes

2.6. Lightweight Cryptography Discussion

Several emerging applications have stringent service quality requirements, such as latency and performance [94]. Conventional symmetric-key encryption algorithms include DES [95], based on Feistel Networks, and AES [96], based on Substitution Permutation Networks. These encryptions require a round function that iterates for a

relatively large number of rounds to enhance security. For example, AES requires ten rounds for a key size of 128-bit, 12 rounds for a key size of 192-bit, and 14 rounds for a key size of 256-bit. Relying on traditional security in these scenarios may cause an overhead in terms of latency and resources. It also comes with cost utilization. An empirical study conducted by Masoud et al. [97] demonstrated how the high-power cost of implementing encryption algorithms like AES, Blowfish, DES, and RC6 is crucial to developing or modifying a lightweight and low-cost encryption solution. They state that the MixColumn operation in the AES algorithm is considered the most costly operation and consumes a vast amount of energy. In some cases, it may be unavailable or expensive to achieve, leading to low uptake of the solution [98].

Several systems and approaches have been proposed to reduce the required computational resources and latency to overcome the limitations of big data encryption. Bansod et al. [99] proposed a new lightweight compact encryption system based on bit permutation instruction group operation. They developed a new hybrid system with more compact results in terms of memory space and gate equivalents in embedded security. The authors frequently noted that conventional algorithms such as DES and AES consume substantial memory and would be tricky to implement in an embedded system scheme.

To address key generation and management issues, Aljawarneh et al. [100] proposed a multithreaded encryption system for securing big data that generates the key from the plaintext. In this work, the encryption algorithm combined the Feistel network, AES with substitution boxes, and a genetic algorithm. First, the input file is divided into several equally sized blocks, and then each block is split into plaintext and key parts. The Feistel network produces a cipher key that is used in the AES component and the genetic algorithm. The algorithm was evaluated using medical-based multimedia big data and compared to existing standard encryption algorithms in terms of runtime and avalanche effect with promising results. Dawood et al. [101] proposed a new symmetric block cipher

model for securing big data. It uses a 512-bit block size and a key length of 128 bits, which can be expanded to up to 512 bits. The cipher supports high key agility and relatively fast encryption speed. However, it is designed with the heavy weight of eight degrees of polynomial equations and three layers of four iterated stages, which makes the encryption a heavyweight process. Lightweight Dynamic Crypto (LWDC)[102] is another block cipher that was proposed to address the speed requirements of modern applications. Encryption and decryption use simple XOR operations followed by substitutions and transpositions along with Cryptographically Secure Pseudo Random Number Generators (CSPRNG) to generate and share the shared value. The cipher outperforms AES in execution time and the CSPRNG puts the cipher on the level of modern symmetric encryptions. For lightweight encryption, Msolli et al. [90] suggested a 5 rounds AES encryption algorithm for multimedia and real-time applications in a wireless sensor network. By reducing the number of encryption rounds, their aim is to reduce the execution time of encryption process. The security analysis histogram of the method shows good encryption and randomness. However, there are many reported attacks [103] [104] on 7 rounds of AES-128 which makes this work critical in term of security and cryptanalysis.

Selective data encryption is considered a way of reducing computing cost while protecting data in clouds. For example, Gai et al. [105] attempt to address the privacy concern that arises from unencrypted transmissions of large amounts of data. They propose a new model that aims to maximise the privacy protection scope by using a selective encryption strategy within the required execution time requirements. To deal with the computation workload caused by large-volume data, this method gives encryption priority to data that carry sensitive information. It uses a Dynamic Encryption Determination (DED) algorithm to dynamically select data packages that can be encrypted under different timing constraints. An encryption technique known as SEEN was developed by Puthal et al. [106] to secure big data streams. This technique employs the selective encryption concept, based on the sensitivity level of the data, to make a trade-off between security,

performance, and resource utilization. The technique also enables the adaptation of different keys based on three levels of data confidentiality, namely, no confidentiality, partial confidentiality, and strong confidentiality, and employs a standard shared key initialized and updated by the data stream manager without the need for retransmission. The method was compared to AES-128 and AES-192 to evaluate its performance and security. The results indicate that the method is faster than the AES-128 and AES-192 protocols while offering the same level of security. To protect medical images, Khashan and AlShaikh proposed a lightweight selective encryption scheme [107] incorporating edge detection, one-time pad, and chaotic map approaches. The approach is based on a single round of encryption that employs edge detection to determine which edge map from an image to use. The original image is then classified into non-blocks of pixels, with the blocks containing the optimal pixel position, based on a threshold value, being considered for the encryption process. Analysis of this method reveals that it is significantly efficient in terms of the time required to encrypt the data and robustness against attacks.

Although selective encryption represents one of the most promising solutions for lowering data protection costs while maintaining adequate data security, the majority of selective encryption algorithms proposed in the literature use static definitions of encrypted parts and encryption parameters [108][109][107]. This property restricts the algorithm's applicability to a small set of applications. It would be ideal to have the capacity to dynamically define the encrypted portion and encryption parameters in response to different applications and requirements.

Dynamic keys have been used in several encryption approaches to overcome multi-round computational complexity [75] [113] [81]. These approaches follow a dynamic structure where the structure of all cipher primitives, such as substitution and permutation tables, changes depending on the dynamic key, which allows a reduction in the number of rounds, leading to a reduction in computational overhead without lowering the security level [110]. A single-round structure cipher to encrypt two blocks at a time was proposed

in [98]. The mode of operation is based on the dynamic key approach, whereby blocks are selected and mixed according to a dynamic permutation table. A similar approach was adopted in [114], where a lightweight cipher schema generated a dynamic key for each input message by hashing the session key.

Chaos theory was recently used in cryptosystem design [112] due to its desirable features, such as pseudo-randomness, complexity, and sensitivity to initial parameter changes [115]. The author in [116] proposed a single round chaos-based image encryption algorithm with orbit perturbation and dynamic state variable selection mechanisms. In this encryption, the orbit of the chaotic map is continuously perturbed by the previously processed pixel using an auxiliary variable, and one of two state variables produces the keystream. Consequently, the keystream relies not only on the chaotic map's initial state and control parameter but also on the information in the plain and cipher images. Thus, the technique is sensitive to both images. Jallouli et al. [117] proposed a stream cipher that uses a combination of multiple chaotic maps for improved robustness, security, and complexity. However, most of these schemas have various limitations, such as vulnerability to classical attacks [118] and complexity of floating computation and hardware implementation [119]. Recently, Ding et al. [120] attempted to overcome the chaos problem by proposing a chaos-based algorithm that utilises a logistic map alongside nonlinear feedback shift registers. The algorithm has been analysed using conventional cryptanalysis as well as a statistics-based experiment and the results were encouraging.

Other studies have attempted to hybridise chaos theory with existing encryption algorithms, such as AES [121] and S-AES [115] and DNA encoding techniques [122]. In these studies, chaos theory was used to produce the encryption keys, or the permutation tables used for encryption. For example, A lightweight encryption algorithm called LCHAOSAES based on AES and chaotic sequences proposed in [123]. This work aims to improve efficiency by reducing the number of AES rounds. In addition, Logistic and Tent chaotic systems are used to generate dynamic keys for encryption in order to make the

algorithm more secure. The results showed a significant reduction in execution time when compared to AES-128. However, there is still a lack of security analysis in this work to prove its strength.

DNA computing has entered the field of cryptography due to the massive parallelism, storage, and ultralow power efficiency of DNA molecules. Several encryption algorithms that combine chaos and DNA computing have been proposed [124] [125]. The core of these algorithms is DNA encoding and DNA computing and contains algebra and biological processes, such as the complementary pairing rule, DNA addition, DNA subtraction, and DNA XOR. In [126], a new encryption algorithm was proposed using Lorenz and Chen's chaotic systems [127]to generate chaotic sequences along with DNA operations. The use of the chaotic system and DNA coding to confuse and diffuse audio files was presented in [128]. The algorithm uses Piecewise Linear Chaotic Map (PWLCM) to generate chaotic sequences and SHA-256 to generate the initial values based on the original audio. To create encrypted audio, the DNA matrix created by dynamic coding is XORed with the key DNA matrix created by the chaotic sequence. Compared with the existing algorithms, the algorithm has a large key space, strong key sensitivity and can resist different attacks.

However, recent cryptanalysis of DNA-based encryption algorithms have revealed security vulnerabilities in some algorithms. For example, the algorithms developed by Liu et al. [129]that used DNA encoding and a 1D Logistic map were broken by Ozkaynak et al. [130] using a chosen-plaintext attack and obtaining the secret key by four chosen plain images. Liu et al. [131] re-evaluated the algorithm's security and found two flaws: the encryption scheme's insensitivity to plain images and the inability to withstand known-plaintext and chosen-plaintext attacks.

Furthermore, the previous DNA-based encryption algorithms have demonstrated that the DNA encryption and decryption rules are the same for different original texts or

images [126] [132] [133]. This situation reduces the algorithm's ability to withstand brute force attacks and chosen-plaintext attacks.

To overcome those shortcomings, dynamic and more complex DNA coding algorithms have been proposed [134] [135][136][137] [138]. They first defined various DNA coding rules before employing chaotic sequences to dynamically select DNA coding rules. Furthermore, some of these algorithms replaced the typical single-base complement process with a complement process, based on the base complementation concept, to increase the complexity of DNA operation [135] [138][137]. These encryption algorithms produced superior encryption results.

The most difficult task when designing a lightweight algorithm is balancing performance, security, and cost. In block encryption, the cost-performance trade-off is given by hardware platforms, the security-cost trade-off is given by the key size of the algorithm, and the security-performance trade-off is given by the number of rounds. In conventional encryption algorithms, such as AES and DES, that use a static S-box utilizing the same S-box in each round, security is ensured by increasing the key size in the cryptographic systems. However, increasing the size of the encryption key is not the optimum solution: no matter the size of the key, its cryptography is ultimately breakable with the proper amount of creativity and persistence [106]. Furthermore, larger keys frequently necessitate more computational resources [107] [109].

Several attempts have been made to improve the security and the performance of conventional encryption by replacing the static S-box with the dynamic key-dependent method [130].

The dynamic key method has been used to propose a set of lightweight cryptographic algorithms [75] [113] [81]. These methods generate dynamic keys as a function of a secret

key and nonce. The encryption structure is varied and unknown to adversaries because of the dynamic key technique. Consequently, this approach introduces randomness and complexity that create obstacles for attackers [139].

The influence of a dynamic key-dependent S-box in cryptographic algorithms was preferred by the authors in [140], leading them to build an algorithm that produces dynamic S-boxes. Elliptic-curve cryptography technology created 16 different S-boxes with a good security architecture that will eventually cause a strong avalanche effect.

Nadu [141] also discovered dynamic key-dependent encryption to improve the algorithm's avalanche effect by using pseudorandom numbers to generate S-boxes. Similarly, Kazlauskas et al. [142] described a method for creating a block cipher system with dynamic S-boxes and reverse S-boxes. It was assumed that any change in the key would result in a fundamental change in the structure of the key-dependent S-box, making the dynamic key-dependent encryption immune to linear and differential cryptanalysis [141].

This literature review reveals that researchers improve the performance of the encryption algorithms by using selective encryptions [105] [106] [107] [108], adding dynamicity for the key generation, or employing the encryption structure using chaos theory or a pseudorandom generator [109] [102] [141]. These approaches help increase security and minimize the overhead or the delay [75] [113] [81]. However, the aforementioned work still exhibits gaps in terms of addressing the simple and lightweight cryptography solution. Furthermore, there is no consideration of using lightweight encryption in privacy-preserving techniques.

2.7. Summary and Conclusion toward Research Gap

The aim of this research is to investigate privacy preserving approaches for the reduction of the privacy risk for data subjects that is associated with collecting, archiving, and transferring personal information.

In seeking this aim, this chapter addressed the following research questions:

1. What are the issues and challenges with existing privacy-preserving approaches?

2. What are the issues and challenges in existing cryptography approaches?

The answer to the first question is grounded in some of the author's published work, including *Rise of Big Data – Issues and Challenges* [43]. This paper focused on the problem of privacy and the techniques that can be used to handle user anonymity. It observed that anonymization approaches for unstructured data have been widely studied in the recent past, and that due to this, various techniques have been developed. However, anonymising unstructured data, mainly text data, is complex and requires more effort compared to the anonymization of structured data. The main challenge in anonymising unstructured data involves finding the sensitive attributes that are dispersed throughout the text.

In this chapter, a critical review of privacy preserving approaches was conducted. This involved discussing the concept of privacy, understanding personal data, reviewing and classifying the existing privacy preserving approaches (based on the purpose of the research) into Privacy Preserving Data Publishing (PPDP) and Privacy Preserving Data Mining (PPDM), and outlining the costs and benefits associated with each approach. PPDP includes anonymization and Pseudonymization.

The review of current privacy preserving approaches for unstructured data through de-identification shows that previous works have been largely developed within the field

of data anonymization, where data are shared with untrusted third parties for secondary use and there is no mechanism by which an individual's identity can be recovered. Pseudonymization is a promising technique to fulfil the requirements of recovering de-identified data while protecting individual privacy. However, existing approaches for Pseudonymization tend to encrypt or remove direct identifiers from the dataset to achieve a certain level of privacy. However, data that could indirectly identify a person may be left in place.

The chapter also reviewed and analysed the new GDPR regulations for processing personal data. The GDPR provides several regulatory incentives to adopt pseudonymization. There are, therefore, significant benefits associated with using it, which include enabling data processing for secondary purposes without the need to obtain the explicit consent of data subjects. However, for this exemption to apply, pseudonymization should meet the GDPR standard, and the existing pseudonymization techniques were developed long before GDPR requirements were established. Many implementations of pseudonymization approaches use static pseudonyms for data subjects [143], while others may contain indirect identifiers; in both cases, these fail to protect against re-identification due to privacy breaches arising from linkage attacks.

To answer the second research question, the chapter examined current cryptography approaches for text data to evaluate their feasibility for application on large scale data. The review indicated that most existing cryptographic systems rely on large key sizes and substantial numbers of rounds to enhance security and reduce the risk of cryptanalysis attacks; this produces overheads in terms of latency and computational resources for big data and real-time applications. Hence, the concept of dynamic keys, which was introduced at the end of this chapter, is notable due to its emphasis on designing an algorithm that relies on the idea of dynamic encryption rather than increasing the key size and number of rounds.

Taken together, this chapter's findings indicate that there is a need for new and lightweight de-identification approaches to detect and encrypt personal data (including direct and indirect identifiers from unstructured text data) to reduce the risk of data subjects and comply with GDPR requirements. The next chapter explains the methodology of the approach that this thesis proposes as a way to achieve the main research objective.

Chapter 3 : Methodology

The primary aim of this chapter is to present a description of the research process. It starts by introducing Design Science Research (DSR) as the methodological basis of research used to achieve the goal of developing a reversible de-identification model for unstructured textual data. This chapter will first describe DSR activities, the concept of creating artefacts and how those DSR activities apply to the activities of the artefacts for this thesis. A justification and rationale for the research approach will then be presented. Finally, this chapter will provide a research design and justification for the data set that was chosen to evaluate the proposed system.

3.1. Research Methodology

Research methodology refers to the process of providing an accurate description of the given problem through a series of stages and steps. Gordana [144] categorised the research methodology of computer science into three categories: theoretical, experimental, and simulation. Whereas Elio et al. [145] classified it into formal, experimental, build, process and model. A formal methodology is widely used in theoretical research to mathematically prove the correctness of algorithms and systems while experimental research methodologies are used to assess new problem-solving solutions.

A build research methodology involves creating an artefact, which may be a physical object or a software device while a process methodology is used to comprehend mechanisms used to complete the tasks. A model methodology is focused on designing an abstract model that is less complex than a real system that it models and is used to gain a better understanding of the actual system as well as to carry experiments that are impossible on the actual system due to the cost and accessibility limitations.

The differentiation between exploratory sciences (traditional) and design sciences was demonstrated by Aline et al [146]. The authors emphasised the significance of creating

science dedicated to the study of man-made artefacts and how to build these artefacts to achieve significant improvements. Over the past few decades, DSR methodology has been used in engineering and information management research to create new innovation and artefacts.

3.2. Design Science Methodology

DSR is a collection of analytical methods that can be used to theorise about an Information System (IS). It aims to enhance the functional performance of the developing artefacts by focusing on the production process and performance assessment of such artefacts on a continuous basis [147]. DSR artefact refer to any IS objects that vary from algorithms, computer interfaces, mathematical models and equations to informal narratives and descriptions in a natural language. Figure 3-1 summarises the processes and outcomes involved; these tasks iterate frequently between the creation of an artefact, its assessment and subsequent feedback to improve the design.

A brief explanation for each process is provided as follows:

1. **Awareness of the problem:** Researchers should define and comprehend the problem to identify required specifications for the system that is being considered. A proposal is the result of this process.
2. **Suggestion:** This process refers to identifying potential solutions to the problem. It is an innovative process in which new functionality is designed using a combination of existing or new and existing components. A tentative design is the output of this phase.
3. **Development:** A tentative design is refined and constructed during this step. The deployment approaches will, of course, differ depending on the objects to be created. An IS artefact is the outcome of this step.
4. **Evaluation:** Once implemented, the artefact is assessed according to criteria defined in a proposal from the awareness of problem phase. Quantitative and qualitative deviations from defined specifications are carefully recorded and tentatively explained.

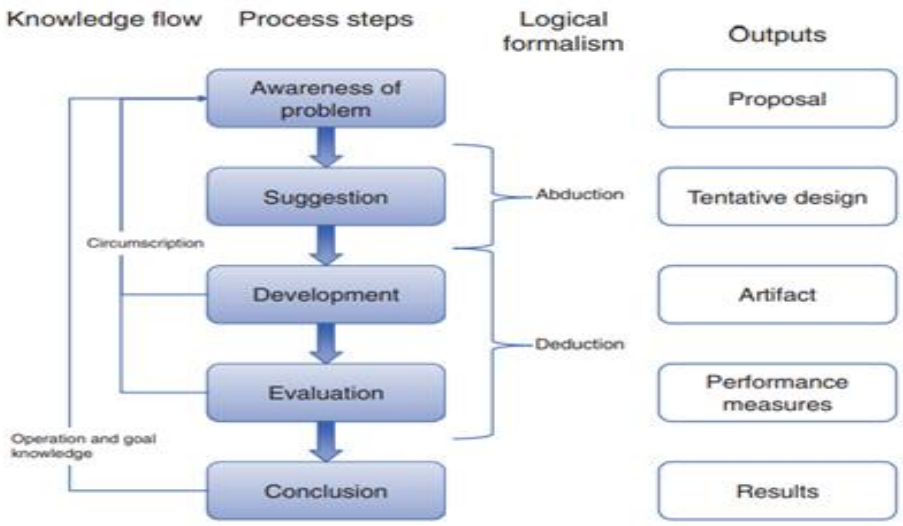


Figure 3-1: Design Science Research Processes and Outcomes [148]

In this thesis, a combination of three methodologies from Elio et al are applied [145]. The formal research methodology is used to mathematically proof the correctness of the proposed artefacts, the build research methodology is used to create the proposed artefacts and a quantitative experimental approach to evaluate them. There are two artefacts created for this work: first, a novel lightweight encryption algorithm that aims to encrypt textual data in a cost-effective manner; and second, a reversible de-identification model ARTPHIL , that aims to de-identify health data to reduce the privacy risk for the data subject.

DSR methodology was chosen as an expansion and guide for these combinations because of the need to iterate more frequently between the development of an artefact, its evaluation and subsequent feedback to further improvements.

- Objective 1: Critically analyse current privacy-preserving approaches for big data.

A literature review was conducted to understand the privacy concept in the information domain, privacy concerns, and the state-of-the-art in existing privacy-preserving techniques. Most importantly, the research identifies the characteristics of the existing techniques and locate the research gap.

- Objective 2: Critically analyse current cryptography approaches for textual data.

A literature review was conducted on cryptography approaches, covering well-known algorithms and their limitations. The criteria for designing a secure cipher and the performance and security evaluation metrics were also considered.

- Objective 3: Design and develop a lightweight encryption algorithm that can achieve a satisfactory level of security with shorter execution times and fewer resources to meet the requirements of delay-sensitive and data-intensive applications.

- Objective 4: Evaluate the developed algorithm against evaluation measures and benchmarked algorithms.

- Objective 5: Design and develop a reversible de-identification model using the developed lightweight encryption algorithm as a replacement strategy to anonymise unstructured textual data.

Next section will demonstrate how the DSR processes used to attain the goal of this thesis as follow:

3.2.1. Awareness of the problem

This was the initial phase of the thesis. It started with reviewing current and related literature to establish the state of the art and define any research or knowledge gaps. The main goal was to investigate current privacy-preserving approaches for big data and identify their weaknesses. The research conducted a comprehensive literature review of existing anonymization and de-identification approaches, cryptography algorithms and their limitations and selection of criteria used to measure the usefulness of the data de-identification models, as discussed in chapter 2.

3.2.2. Suggestion

After studying the literature, the objectives linked to the fundamental aspects of security, privacy and performance is formulated. This thesis aims to contribute to the knowledge by designing a reversible de-identification model that can achieve individual privacy with shorter execution times and fewer resources. The following research objectives were suggested to accomplish this goal:

1. Design and implement a lightweight encryption algorithm.

2. Use the latest advanced information extraction algorithm to detect and encrypt personal information from unstructured textual data.

3.2.3. Development

In this stage, the research designed a proposed algorithm, which was named E-ART, based on the concept of balanced binary tree and the American Standard Code for Information Interchange (ASCII) values. The design of the proposed E-ART algorithm was divided into three phases: substitution method, adding offsets and dynamic key construction, as discussed in chapter 4 E-ART was implemented on a Java platform using NetBeans 8.2, an open-source integrated development environment (IDE) with its

accuracy empirically proven using 100 unstructured clinical notes, as discussed in chapter 5.

This research also designed and implemented ARTPHIL, a reversible de-identification for the de-identifying of a free-text model. ARTPHIL consisted of two key components that are integrated to de-identify unstructured textual data: the core of the Philter package [149] a state-of-the-art tool for extracting personal identifiers from free-text to detect confidential information, and the E-ART encryption algorithm [84] for replacement strategy. ARTPHIL was implemented using Python 3.7 (32 bit) platform due to the strength of Python libraries for natural language processing (NLP) that helps to detect PHI entity.

3.2.4. Evaluation

In this phase, the developed artefacts have been evaluated against the relevant security and performance criteria. The evaluation process was carried out twice, first with the E-ART algorithm and then with ARTPHIL. A brief explanation for each process in the following sections

E-ART Evaluation

To evaluate the proposed E-ART algorithm, the research conducted a comparative analysis of the performance of E-ART algorithm with benchmarked symmetric encryption algorithms AES-128 and DES algorithms using different file sizes. AES and DES are classical and well-known cryptographic algorithms and many encryption algorithms are driven by their structures [150], [151]. AES is based on the substitution-permutation network structure and was extensively tested for possible loopholes in security before its release. So any algorithm compared to its security level is supposed to be secure. DES is based on the Feistel structure that uses the same code for encryption and decryption, which could contribute to lower memory usage [83].

The parameters used to assess the performance were as follows:

- Processing time: the amount of time consumed during the encryption or decryption process.
- Memory consumption: the amount of memory consumed during the encryption or decryption process, measured in Megabytes (MB).

Security parameters used to assess E-ART were as follows:

- Avalanche effect
- Frequency analysis
- Bit Independence Criteria (BIC)
- Randomness verification using the NIST statistical tests.

Performance was compared in terms of processing time and memory consumption using different file sizes. Security was assessed through the avalanche effect, frequency analysis and NIST statistical tests.

ARTPHIL Evaluation

The research used the following parameters to evaluate the performance of the de-identification system, ARTPHIL:

- Precision, recall and F-measure: recall was selected as the primary evaluation measure since the system was optimised to maintain maximum individual privacy. The recall represented a percentage of PHI entity that were correctly identified. The second evaluation measure was precision, which represented the non-PHI that the proposed model retained since a good de-identification

system should retain as much non-PHI as possible. The F2 measure is calculated in addition to F1, which weighed recall two times higher than precision, to emphasise on sensitivity.

- Execution time: The run time of the ARTPHIL model is computed using one batch and 20 batches of 514 notes using the Python Time function, 'time', to estimate the feasibility of running ARTPHIL on a large scale.

- Re-identification risk: The re-identification risk of ARTPHIL for each PHI entity is estimated by calculating the conditional probability of a leak in a piece of identifiable information.

3.3. Justification of the Research Method

This research is adapted from a design science methodology to develop an artefact that could achieve the research objectives. Research objectives were defined based on critical literature analysis of existing studies, which provided a justification for the development of a new reversible de-identification model for unstructured textual data. Regarding a replacement strategy for the de-identification model, a new lightweight encryption algorithm was developed and tested. A quantitative experimental approach was used in accordance with scientific techniques to obtain empirical evidence to prove the correctness of the proposed algorithm and evaluate the performance of the developed artefact against defined evaluation measures. This quantitative research approach helped to interpret statistical performance, security analysis and provided a useful comparison with existing algorithms. The design cycle (design, build and evaluate loop) within the design science methodology helped us to improve the performance and strength of the proposed model.

3.4. Rationale for Research Approach

Data security and privacy have been recognised as world-wide problems in new and emerging technologies. A detailed literature review on privacy and data security in the domain of unstructured textual data revealed that there is limited research on a reversible de-identification system that generates an anonymized version of data with options for reversing the anonymization and recovery of the original data, if needed (objective 5). Additionally, the operational cost, such as processing time and memory consumption, should be maintained, particularly for delay-sensitive and data-intensive applications. However, it can be challenging to maintain a balance between the performance and reliability of an algorithm [94] [114].

A literature review of current privacy-preserving approaches provided evidence on the importance of encryption for reversible anonymization while a review of existing encryption algorithms focused on security concerns without considering the performance aspect. Many robust cryptography solutions face the limitation of consuming a significant amount of computing resources, such as a high execution time and large memory usage, which proved unsuitable for constrained devices or real-time data processing. For example, the well-known AES encryption algorithm uses a substitution process that iterates for multiple rounds to hide a relationship between the key and the text called (Diffusion). However, these iterations are time-consuming and resource-intensive.

Furthermore, the cumbersome key management and distribution of the traditional encryption algorithm prevent a suitable level of scalability. In this thesis, a more lightweight and practical alternative (objective 3) is design, developed and then evaluated and compared its performance to AES and DES encryption algorithms (objective 4). Then, the proposed E-ART is used as a replacement strategy to develop the reversible de-identification system, ARTPHIL that overcome some of the limitation of existing privacy preserving approaches discussed in chapter 2. Regarding the extraction of sensitive data, the advancement of rule-based and natural language processing (NLP) were used to extract

PHI from unstructured textual data. The proposed system was found to be suitable for protecting individual privacy and reducing information loss caused by irreversible anonymization.

3.5. Research Design

The research investigated the issue of privacy preservation approaches for unstructured data in five phases. Phase 1 involved three stages: Stage 1 was a detailed review on privacy preservation approaches for unstructured big data challenges; Stage 2 was a review of existing cryptography approaches to secure textual data; Stage 3 was identifying research gaps and setting up the research objectives.

Phase 2 was an iterative phase of E-ART algorithm designing, implementing and testing. In Phase 3, the performance and security of the proposed algorithm were compared to benchmarked symmetric encryption algorithms, AES-128 and DES, in terms of processing time and memory consumption. Security was assessed through the avalanche effect, frequency analysis and the NIST statistical tests [152].

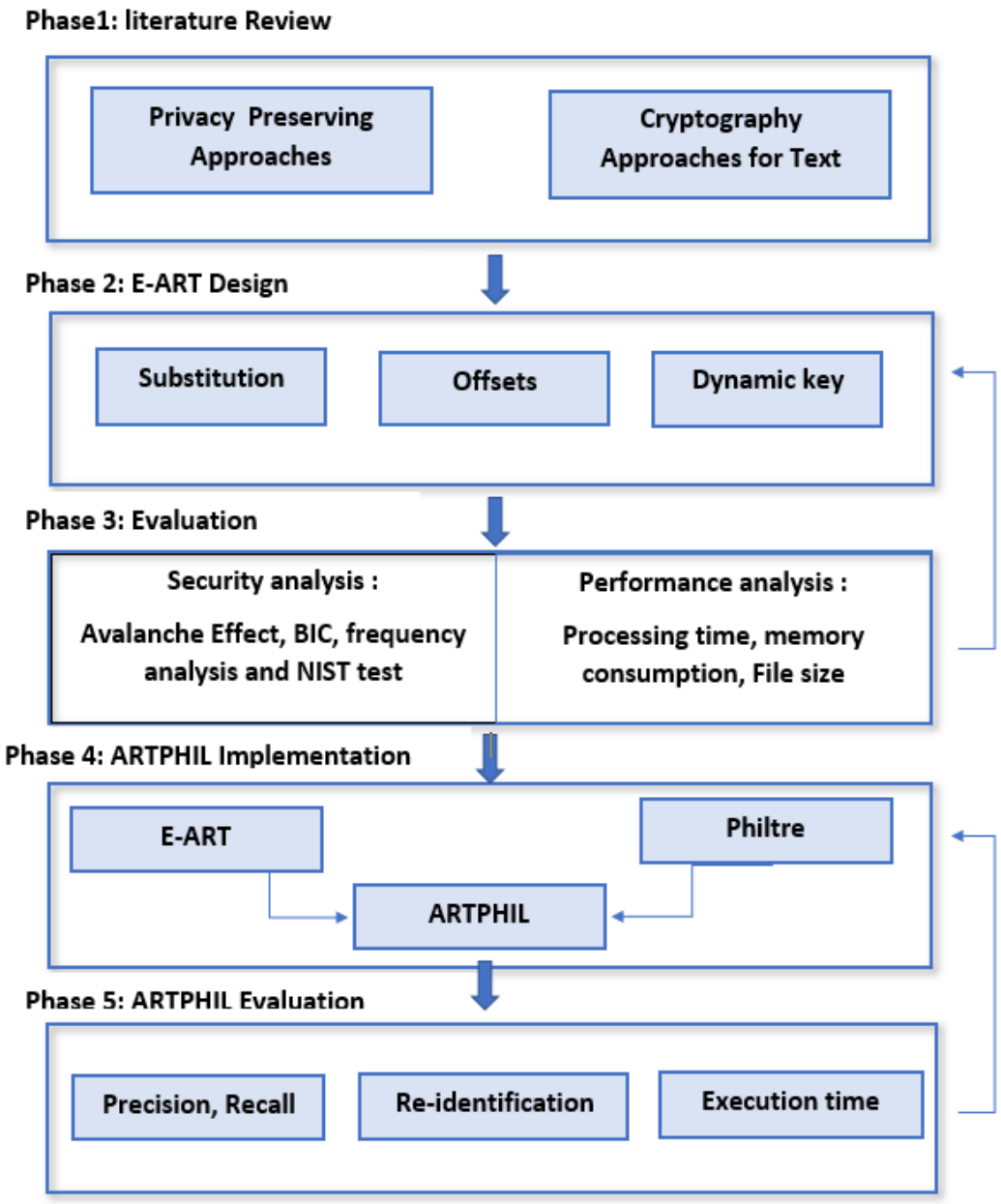


Figure 3-2: Research design

Phase 4 was an iterative phase of the design and implementation of ARTPHIL, a reversible de-identification system through integration of E-ART with the core of Philter [149], the state-of-the-art tool for extracting personal identifiers from free-text. Phase 5 was the validation and evaluation phase of ARTPHIL by running an evaluation script that automatically compared de-identified notes to annotated i2b2/UTHealth 2014 de-

identification corpus at the character level to quantify PHI detection performance and re-identification risks. Figure 3-2 provides further clarification on all five phases.

3.6 Dataset

The main goal of this thesis is to develop a reversible de-identification system that de-identifies unstructured big data in a cost-effective manner. The healthcare domain was used as a case study due to several characteristics that made it suitable for performance evaluation of the proposed system. These characteristics include sensitivity, diversity, volume and usefulness. A brief explanation for each characteristic is as follows:

- **Sensitivity:** Health dataset includes confidential information that, if revealed, may have a negative impact on the data subject. Consequences from disclosure may include social stigma, financial loss, racism, mental and emotional distress [153]; thus, privacy is required to create public trust for individuals to successfully engage in a specific activity,

- **Data diversity:** Medical data comprises of various data structures such as unstructured free-form clinical notes, structured and unstructured textual data, photos, sounds and videos [153]. This makes data protection even more complicated, especially when using a uniform algorithm across different structures. Therefore, this research is aimed at anonymising unstructured textual data.

- **Volume**: Healthcare is a data-intensive field that generates vast volumes of data from various sources, including hospitals, primary care practices, clinics and laboratories [154]. This is due to the existence of non-interoperable programmes and patients' nomadic nature in seeking treatment from different medical facilities for various reasons. Furthermore, medical records must be retained for the duration of an individual's life [154] and unlike data from other

fields, healthcare data remain valuable over time. This distinctive characteristic poses a challenge for the preservation of healthcare data.

- **Usefulness:** Medical textual data provide plenty of opportunities for a variety of fields such as research, public health and legislation to learn and generate new information and insights for managing and enhancing healthcare services. Employers and insurers require this data to analyse treatment expenses and pay associated bills. The same knowledge can also be used for personal economic benefits in areas such as drug promotions, bank lending decisions and employers' hiring decisions. However, the usage of textual data puts a strain on a patient's privacy, which increases the need for a different de-identification system to suit various circumstances [155] [156].

This research used the i2b2/UTHealth 2014 de-identification corpus for the NLP Shared Tasks Challenges that was released by [157] as part of the i2b2 National Centre for Biomedical Computing, whose de-identification guidelines reported by conformed to the safe harbour criteria. The PHI in the data set was hand-labelled and obscured and substituted with the appropriate surrogate before release. The data sets comprised of 1304 longitudinal clinical notes of 296 patients with 2–5 notes chosen per patient and were officially divided into training and testing sets. The training set consisted of 790 documents (including 269 for validation) while the testing set contained 514 documents. Each document is a clinical note in XML format and the PHI entities within the documents are annotated as text spans with corresponding entity types.

The typical dataset size used in medical research varies depending on the factors that need to be tested or examined. For example, the study conducted in [158] to investigate the correlation between Statin use and the risk of cancer examined 25,811 health records while the research conducted in [159] use free-text clinical records to identify the reason of retirement use only 5910 records. However, a typical dataset to purchase from a medical

repository such as CRPD Gold [160], to run epidemiological studies would be around 15,000-30,000 of patients' records. Then this set would be spliced in various ways for different subsets of interest such as gender [161] and age group [162], with around 500-600 records in each batch.

The goal of developing a de-identification system is to make data available for research without compromising privacy. Many types of research in de-identification systems [163][164] have been tested against performance metrics such as precision, recall, and f-measure, but the execution time is not considered dominantly.

In this research, the execution time was calculated in addition to performances metrics in order to test the feasibility to run the proposed system on large scale of data. The run time of the integrated model was calculated using single batch of 514 free text notes and with 20 batches of the 514 notes, resulting in 10,280 notes overall.

3.7. Summary

This chapter explains how the research was conducted to attain the goal of this thesis which is to develop a lightweight de-identification model for individual privacy. It describes a DSR methodology for creating knowledge through the production of artefacts while explaining how they pertain to the artefacts at the core of this thesis. Two major design science research artefacts helped achieve the research objectives. ARTPHIL is a novel de-identification system that anonymises unstructured textual data with an option to recover the original data. E-ART is a lightweight encryption algorithm that used as a replacement strategy in ARTPHIL system. Both artefacts have been evaluated against the defined evaluation metrics.

This chapter also justifies of choice of the research methodology and a rationale for the research approach while describing the data set chosen to evaluate the artefacts in this research and the reasons for that choice.

Chapter 4 : E-ART Design

As explained in the previous chapters, data privacy and security have become significant issues due to the increasing applications and capabilities of big data. Although robust cryptographic solutions are available, as discussed in chapter 2, their application in the face of ever-increasing volume, variety, and speed remains challenging [115]. Most existing cryptographic systems rely on increasing the key size and the number of rounds to enhance security and reduce the risk of cryptanalysis attacks. However, this effort causes overheads in terms of latency and computational resources for real-time applications. For example, the AES [165] requires several iterations over a round function, which negatively impacts the system's performance. As a result, many current applications have abandoned data encryption as a mechanism to reach an adoptive performance level [105]. Therefore, there is a need for an effective cryptographic algorithm that can fulfil big data requirements, such as speed and security, in a cost-effective manner. This chapter focuses on designing a new and alternative cipher scheme called E-ART that aims to overcome the disadvantages of existing ciphers, such as large key and complex computation. The overall goal is to achieve a satisfactory level of security with shorter execution times and fewer resources.

The chapter is organised as follows. Section 4.1 describes the stages of designing the proposed algorithm. Sections 4.2 and 4.3 summarise the encryption and the decryption process, respectively. Section 4.4 discusses the proof of correctness of E-ART, and section 4.5 summarises the chapter.

4.1. Algorithm Design

The main novelty of the E-ART algorithm is its use of the reflection property of a balanced binary tree data structure to enhance data search efficiency. The design of the

algorithm involved three stages as shown in Figure 4-1. The research first designed a preliminary function for the Substitution Method. Then added two offsets, via Stages 2 and 3, to solve the issues that occurred in Stage 1.

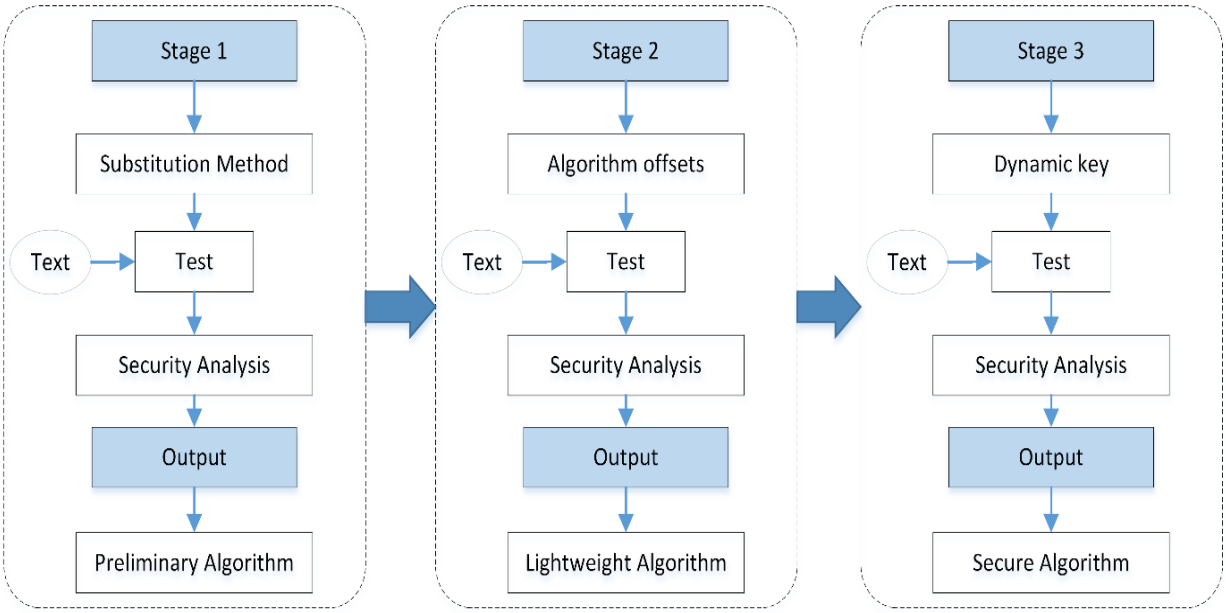


Figure 4-1: Algorithm Design Process

4.1.1. Stage 1: Substitution Method

In Stage 1, a Novel substitution method was proposed based on the concept of the reflection of a balanced binary tree along with along with American Standard Code for Information Interchange (ASCII) values of text characters to encode data. Binary trees can be explained as follows; The root node has a value *X*. The left subtree of the main tree contains all values less than *X*, while the right subtree includes values greater than *X*, so there is no need to visit every node when searching for a specific value. Thus, the binary tree has a search complexity of *O(log(n))* when searching for a particular value. Hence, the binary tree can enable great efficiency.

To explain the proposed method, Table 4-1 presents the ASCII table for the character from 0 to 127. The table contains three clusters of five columns apiece. The first column contains the decimal value (Dec.), the second column the hexadecimal value (Hex), the third column the binary value, the fourth column the octal value, and the fifth column the character value that displays on the screen. In the proposed substitution method, the focus was on mapping from the decimal to the character value.

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

Table 4-1: ASCII Table

Figure 4-2: Binary Tree of E-ART illustrates a partial version of the E-ART balanced tree; the complete version can be found in Appendix IV. The tree nodes represent the ASCII Decimal values, ranging from 0 to 127. E-ART supports the English characters by mapping each ASCII Decimal value from the ART tree to its equivalent character in the ASCII table. In Figure 4-2 E-ART has five levels. Node 64 is the Root, with children 32 and 96; node 32 has two children, as does node 96, and so on. Each branch of the tree can be defined as a link between parent-child nodes, e.g. the branch from 64 to 32, or the branch 88-92.

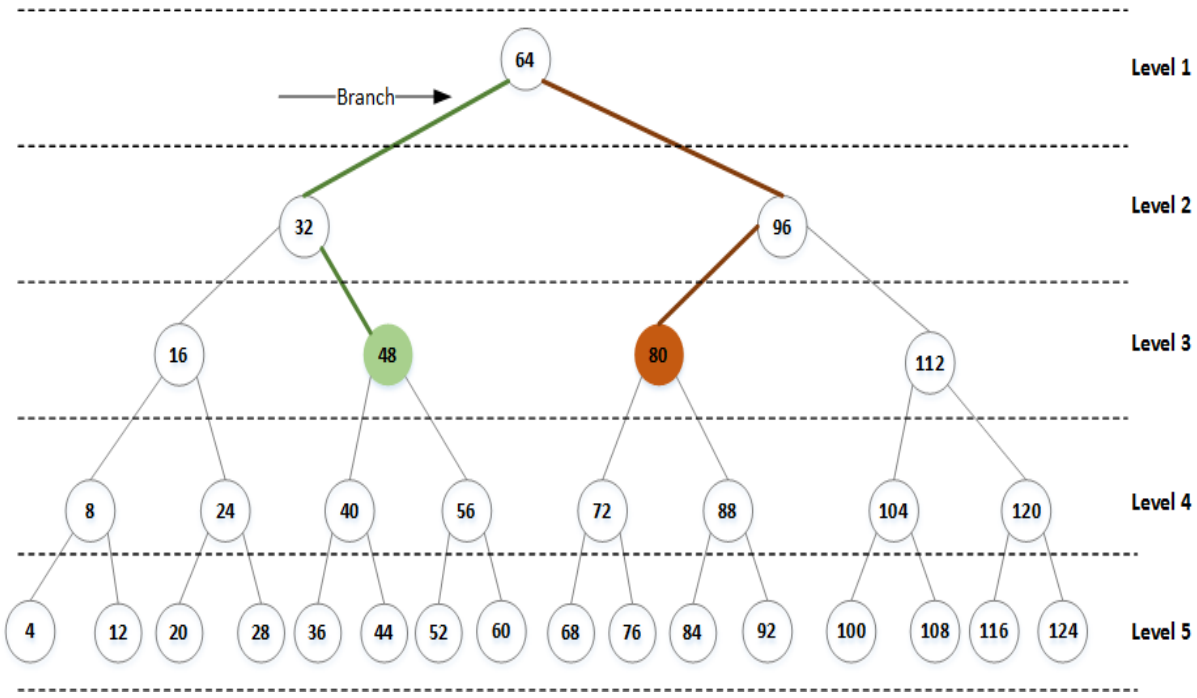


Figure 4-2: Binary Tree of E-ART

Let us consider a text containing the character "P," whose ASCII code is 80. This character is on level 3 of the binary tree and is located in one right and one left branch (1R1L) of the root node. Its reflected character is at 1L1R, which is 48, as shown in Figure 4-2 In the ASCII table, 48 is the code for the character "0," so the initial reflected value of the character "P" is 0. Thus, any character in the plaintext can be encoded using a reflection tree. As a result, a technique with higher search efficiency is achieved. For a balanced binary tree, as shown in Figure 4-2: Binary Tree of E-ART, the initial reflected value $Val_{initialref}$ can be computed from the original value Val_{org} as follows:

$$Val_{Initial\ ref} = (Len_{max} - Val_{org}) + 1 \quad (4\text{-}1)$$

The proposed Substitution Method is a bijective mapping in which each element of the right side of the tree is paired with exactly one element of the left side of the tree. Each number from the tree has only one equivalent ASCII character which satisfies $(x) \neq (y)$ for $\forall\ x, y, x \neq y$.

However, there are some issues with this mapping that can be illustrated with an example. Consider encoding the message "HELLO World" using the above substitution method. First, each character will convert to its equivalent ASCII value. Then the reflectance value is computed and mapped to the equivalent ASCII character as shown in Figure 4-3.

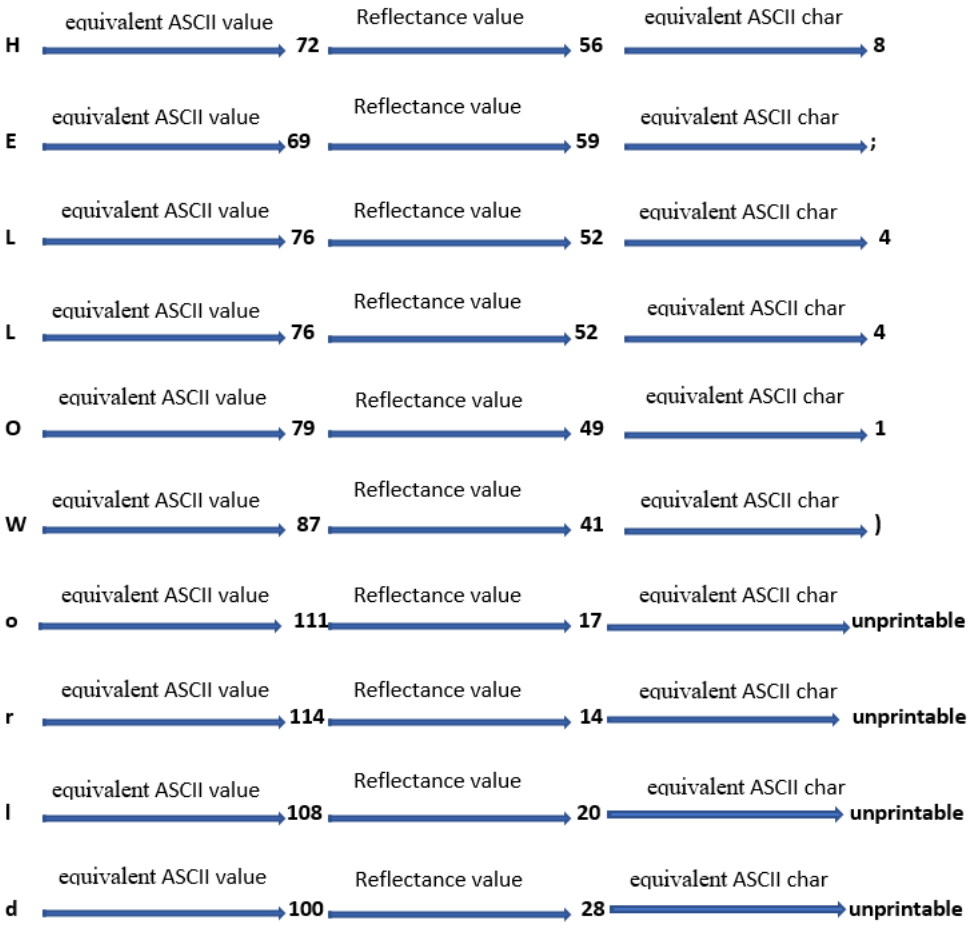


Figure 4-3: Illustration of the Substitution Method (Stage 1)

As demonstrated via the example in Figure 4-3, the following issues occurred and were addressed in Stage 2:

- The presence of unprintable characters, such as space, carriage return, and other text formatting characters, which ranged from 0 to 32 in the ASCII table, was not addressed.

- It is vulnerable to cryptanalysis attacks, such as frequency analysis attacks [121], due to the one-to-one mapping of characters.

4.1.2. Stage 2: Adding offsets

To overcome the issues that occurred in Stage 1, the research proposed a solution of adding two offsets to the initial reflected value: constant offset $Offset_{const}$ and variable offset $Offset_{var}$. All the notions are shown in Table 4-2: Summary of the Notations.

First, to avoid the appearance of non-printable ASCII characters, a constant offset value of 32 is added to the initial reflected value. For example, in the previous example shown in Figure 4-2: Binary Tree of E-ART, the character "o," whose ASCII code is 111 and initial reflected value based on equation (4-1) is (127 – 111) + 1 = 17, represents a non-printable character. However, the newly calculated reflected value after adding the constant offset is 17 + 32 = 49, which represents the character '1'. Therefore, the algorithm does not consider encrypting any unprintable character from 1-to 31 to maintain the bijective property.

Table 4-2: Summary of the Notations

Symbol	Definition
Len_{max}	Maximum ASCII character range (default is 127)
Val_{org}	Original ASCII character value
$Val_{initial\ ref}$	Reflected value of the original ASCII character value
$Offset_{const}$	Constant offset value used to avoid non-printable characters (32 by default)
Val_{ref}	Reflected value of the original ASCII character value after adding the offset
$Offset_{var}$	Variable offset computed based on the properties of the tree
R	The root node of the tree

N_L	Adjusted value of *N* that is within the range of maximum value
N_R	Reflection value of N_L
Pseudo	Pseudo-random number generated using character's position in the text
Variance	An integer number which represents the second part of the secret key that used to adjusted the Pseudo
$Character_{value}$	Equivalent ASCII character for a given value

Second, to prevent a cryptanalysis attack and add more complexity, the research proposed adding another offset, which called variable offset $Offset_{var}$. It is computed based on the E-ART tree's properties and the secret key. It is explained in more detail in Stage 3. The general equation for the reflected value after adding the two offsets is

$$Val_{Ref} = \begin{cases} (X \,\%\, Len_{max}) + Offset_{const} & , X > Len_{max} \\ X & , X \leq Len_{max} \end{cases} \quad (4\text{-}2)$$

where $X = Val_{intial\ ref} + Offset_{var} + Offset_{const}$.

If the value *X* is less than the maximum character space Len_{max}, then it is considered the reflected value. If the reflected value is greater than Len_{max}, then it is computed in the range of $\{0, Len_{max}\}$, and the constant offset is added to avoid the appearance of non-printable characters.

4.1.3. Stage 3: Dynamic key

In Stage 3, the key derivation process was established. The process involved incorporating the proposed dynamic offset and improving the variable offset that suggested in Stage 2. The main objective of Stage 3 was to introduce randomness to the encryption process to provide adequate protection against classical and modern cryptanalysis attacks by making the process difficult to breach without increasing the execution time.

The proposed algorithm uses an initial 128-bit symmetric key that used to generates two offsets, dynamic and static. The static part changes for every session, and the dynamic part changes for each character. The key derivation process is explained in the next section.

Key derivation

In the key derivation process, the initial key is used to construct two offsets that are applied during the data encryption process. The first one, called the variable offset, remains static during the encryption/decryption process. The second one, which is a dynamic offset, changes with each character value. The inclusion of the dynamic offset ensures considerable randomness in the key, thus guaranteeing the security of the encryption process. The construction of the two offsets starts with the initial key.

Initial Key

The initial key is a secret key shared between legal entities that can be renewed after each session or depending on the system's configuration. However, key management between legal entities is not the focus of this work. An initial key consists of two values, *N* and *Variance*, both of which are used to generate the offsets. *N* represents an integer value that can be set to be 64 or 128 bits. This value is the input to calculate the variable offset. *Variance* is used to compute the pseudo-random number value for the dynamic offset. The offsets used in this case are as follows:

- The variable offset is calculated mathematically using the proposed tree properties. The left and right nodes are shown in Figure 4-1. It uses the *N* value derived from the initial key to calculate N_L and its reflection node N_R and then generate the value of $Offset_{var}$. This value is added to the initial reflection value according to eq. (4.1) and (4.2) to add more complexity and prevent cryptanalysis attacks that would take advantage of one-to-one mapping. It is computed as follows:

$$N_L = N \; mod \; Len_{max}$$

$$N_R = (Len_{max} - N_L) + 1$$

$$Offset_{var} = \begin{cases} \mathrm{R} \times N_L \bmod N_R & \text{if } N_L < \text{Root.} \\ \mathrm{R} \times N_R \bmod N_L & \text{if } N_L > \text{Root.} \end{cases} \quad (4\text{-}3)$$

The Dynamic Offset

It is produced automatically using a pseudo-random number and the second part of the initial key (*Variance*). The pseudo-random generator uses each character's position in the text as a seed to generate a pseudo-random number of 64 or 128 bits. The pseudo-random number is then adjusted using the *Variance* value. This offset is added in the last step to produce the final encrypted characters and is changed for each character. This results in a high degree of robustness and resistance to known powerful attacks. The dynamic offset is calculated as follows:

$$\text{Dynamic offset} = (\text{Pseudo}) \bmod \text{Variance.} \quad (4\text{-}4)$$

The dynamic offset is produced as a function of the initial key, the character's position, and the pseudo-random number. The relation between the plaintext and the ciphertext is therefore more random and complicated, which strengthens protection against cryptanalytic attacks since the encryption/decryption process in turn becomes dynamic and different for each character. Combining eq. (4.2) and (4.4) yields eq. (4.5) as follows:

$$Val_{Ref} = Val_{Ref} + dynamic \; offset \quad (4\text{-}5)$$

4.2. Encryption Process

E-ART can be classified as a character-oriented cipher where each character is replaced with another character based on the suggested reflection-balanced tree substitution method and the three offsets described in the previous section. The

substitution incorporated in the algorithm is secret key dependent, non-linear and poly-alphabetic as shown in Figure 4-4. The character 'a' substituted with different characters at the output data.

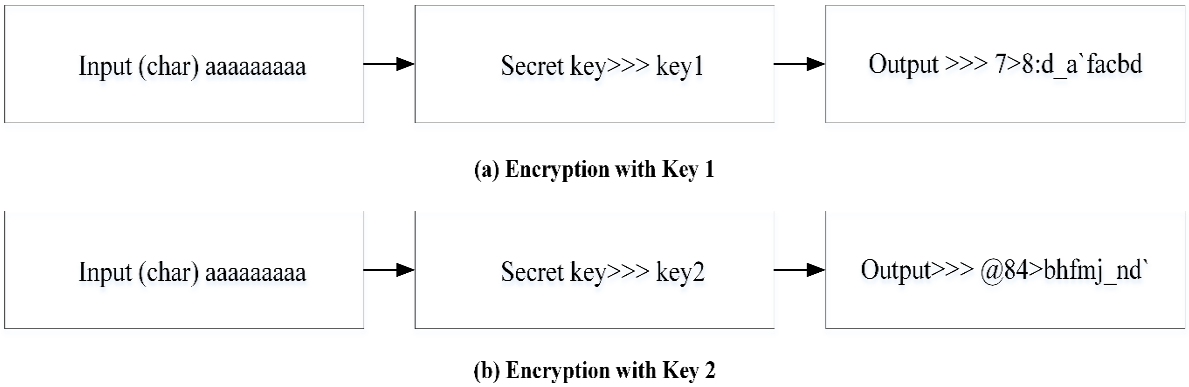


Figure 4-4: Encryption with Different Keys

Figure 4-5 shows a conceptual overview of the encryption process and the pseudo-codes are shown in Algorithm 1 and more details of how the algorithms work can be found in Appendix I. The main steps of Algorithm 1 is described as follows:

- Initially, the input textual data are stored in an array of characters (plaintext list).
- Each character in the list is converted into its corresponding ASCII values and stored in Val_{org}.
- The variable offset $Offset_{var}$ is generated using N, the first value of the initial key, and the properties of the tree – R, N_L, and N_R – as shown in eq. (4.3).
- For each character in the list, the initial reflected value $Val_{Initial\ ref}$ for each character is calculated using eq. (4.1).
- For each character in the list, the dynamic offset is generated by a pseudo-random generator using *Variance* the second value of the initial key and characters' positions, as shown in eq. (4.4).

- Then, value X is generated by adding the variable offset $Offset_{var}$ and constant offsets $Offset_{const}$ to the initial reflected value using eq. (4.2).
- Value X changes based on the maximum length (Len_{max}) and non-printable character range. If the value of X is greater than Len_{max}, then apply the mod operation and then add $Offset_{const}$, as shown in eq. (4.2).
- Then, the dynamic offset is added to the value of X using eq. (4.5) to generate the final reflection value Val_{Ref}.
- Val_{Ref} is converted to the equivalent ASCII character to produce the encrypted character.
- Append the character to encrypted list.
- Once all characters in the plaintext are encrypted, the encrypted text file is generated.

ALGORITHM 1: E-ART Algorithm

Input: *R,* $Offset_{const}$*,* Len_{max}*, input_text, N, Variance*
Output: *Encrypted text*
Initialisation
input_list = Read all words from input file
Get the Val_{org} for each character
Get the $Offset_{var}$ from eq. (4.3)
while all words in *input_list* are not iterated, **do**
  word = pop word from *input_list*
    for each *character* in *word* **do**
      Get $Val_{Initial\ ref}$ of *character* from eq. (4.1)
      Get Dynamic offset from eq. (4.4) with
        Let X = ($Val_{Initial\ ref}$ + $Offset_{var}$ + $Offset_{const}$)
        if X is greater than Len_{max}
          Val_{Ref} = [*(X mod* Len_{max}*)* + $Offset_{const}$ + Dynamic offset]
      else
          Val_{Ref} = [*X* + Dynamic offset]
      end if
    end for
    append $Character_{value}$ of Val_{Ref} to *EncryptedWord*
      append word or *EncryptedWord* to *EncryptedList*
end while
  Write all values from *EncryptedList* to output document

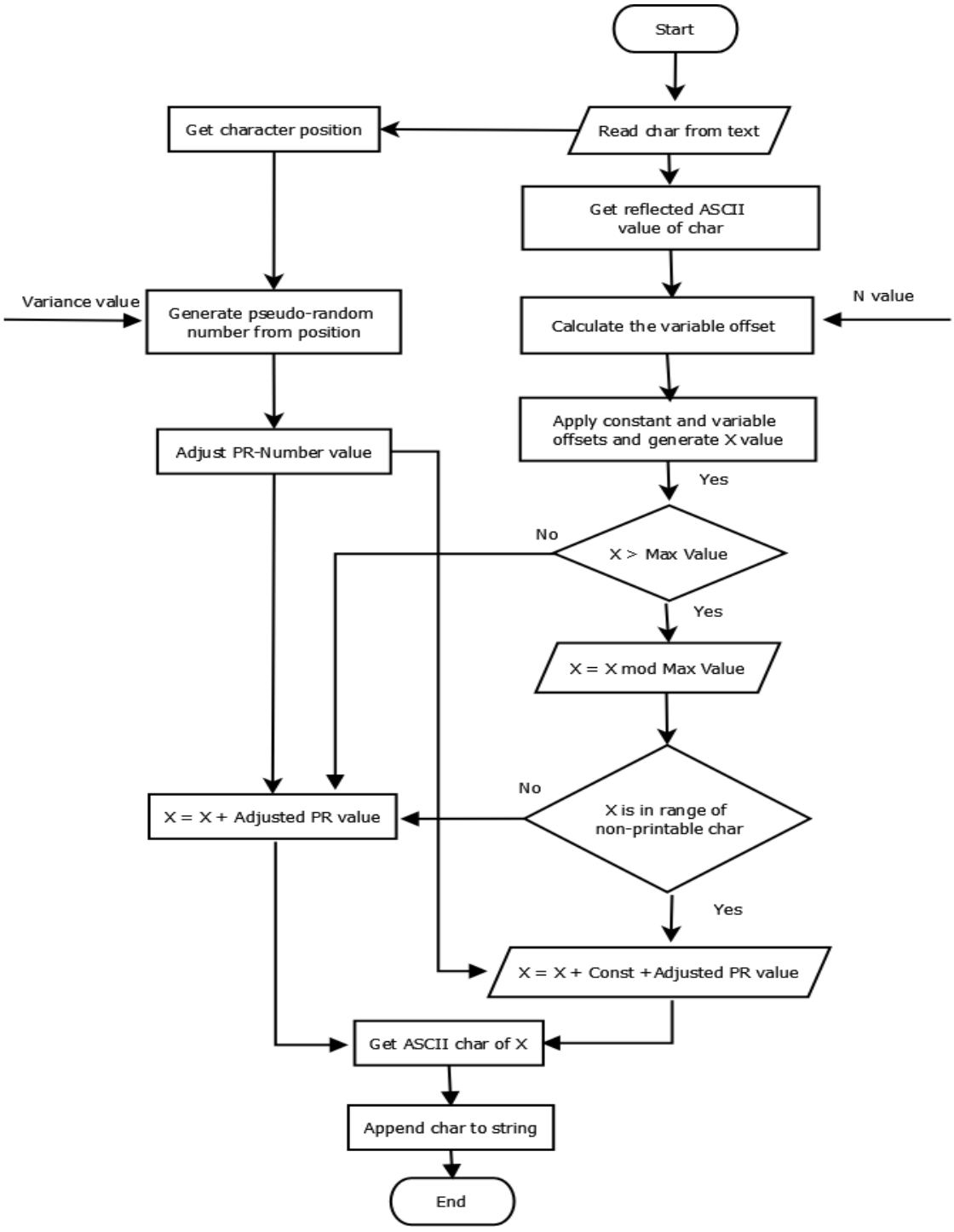


Figure 4-5: Conceptual Overview of the Encryption Process

4.3. Decryption Process

The decryption process can be considered a reversible form of the encryption algorithm. Figure 4-6 shows a conceptual overview of the decryption process. The main steps of the decryption process in Algorithm 2 are as follows:

- Initially, the input data are stored in an array of characters (ciphertext list).
- Each character in the list is converted into its corresponding ASCII value and stored in Val_{org}.
- The $Offset_{var}$ is generated using *N*, the first value of the initial key, and the properties of the tree – R, N_L, and N_R – as shown in eq. (4.3).
- For each character in the list, the dynamic offset is regenerated by a pseudo-random generator using the same parameters, *Variance*, the second value of the initial key and the characters' positions as shown in eq. (4.4).
- Value X is generated by subtracting the dynamic offset from Val_{org}.
- Then, we check: if subtraction of variable offset $Offset_{var}$ and constant offset $Offset_{const}$ from X is less than 0, then set Quotient to be equal to 1; otherwise, set Quotient value to be equal to 0.
- Generate the Val_{Ref} by multiplying Len_{max} and Quotient and then subtract X, variable offset $Offset_{var}$ and constant offset $Offset_{const}$.
- Generate decrypted value by subtracting Val_{Ref} from Len_{max} plus 1.
- Decrypted value is converted to the equivalent ASCII character to produce the decrypted character.
- Append the character to the decrypted list.
- Once all characters in the ciphertext are decrypted, the decrypted text file is generated.

ALGORITHM 2: Data Decryption Algorithm

Input: *R,* $Offset_{const}$, Len_{max}, *Encrypted Text, N, Variance*
Output: *decrypted text*

Initialisation
input_list = Read all word from input file
Get the Val_{org} for each character
Get the $Offset_{var}$ from eq. (4.3)
while all words in *input_list* are not iterated, **do**
    word = pop word from *input_list*
        for each *character* in *word* **do**
            Get Dynamic offset from eq. (4.4)
            Let *X* = Val_{org} – Dynamic offset
            if (*X* – $Offset_{var}$ – $Offset_{const}$) < *0*
                Quotient = 1
            else
                Quotient = 0
            Val_{Ref} = [*(*Len_{max}*x Quotient + X)* – $Offset_{var}$- $Offset_{const}$]
        decrypted_value = (Len_{max} – Val_{Ref}*) + 1*
            If (*decrypted_value >= 0 && decrypted_value <= 32)*
                decrypted_value= decrypted_value + $Offset_{const}$
            end if
        end for
            append $Character_{value}$ of *decrypted_value* to *decryptedWord*
            append word or *decryptedWord* to *decryptedList*
end while
Write all values from *decryptedList* to output document

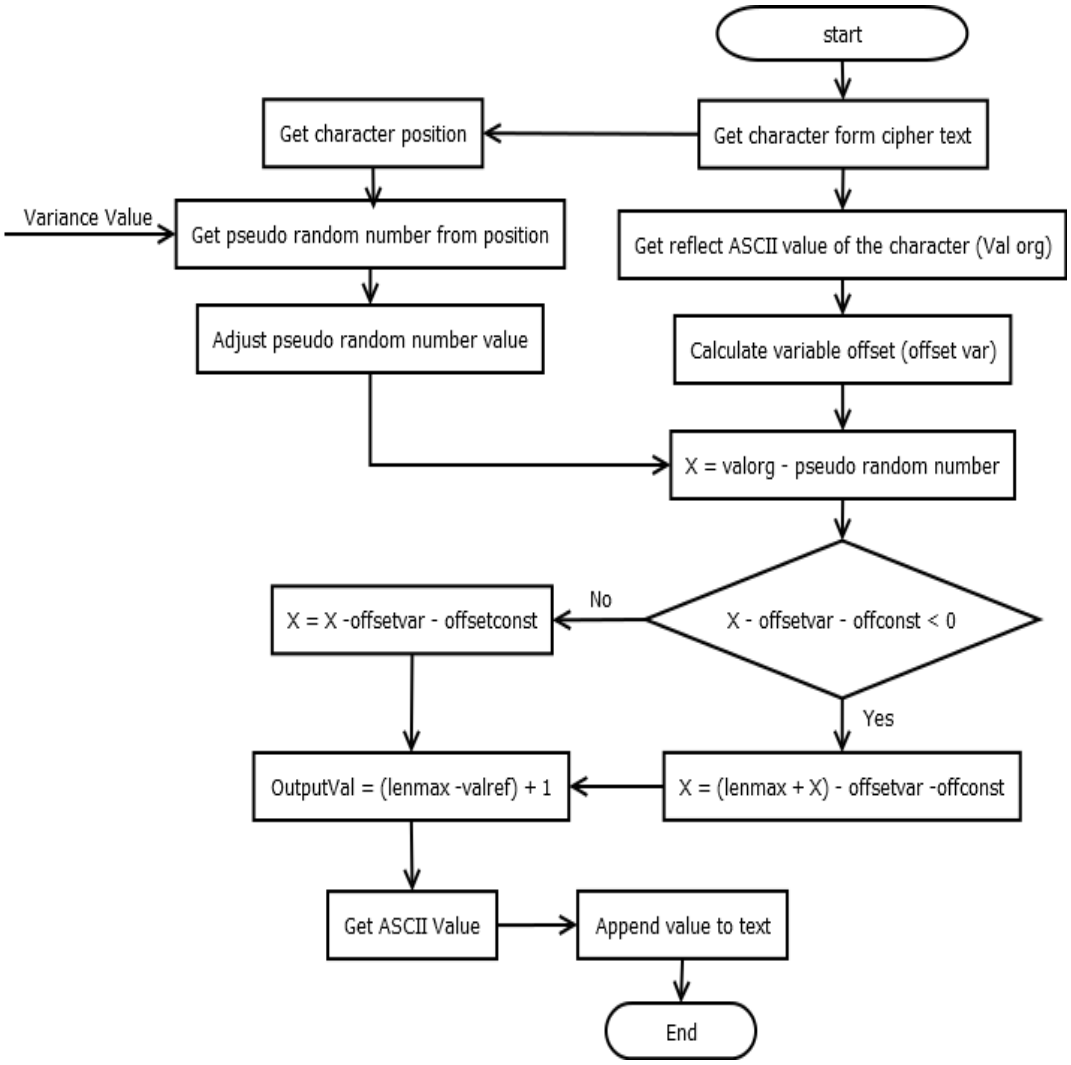


Figure 4-6: Conceptual Overview of the Decryption Process

4.4. Proof of Correctness

In this section, a proof of the proposed algorithm's correctness is provided from an empirical and mathematical perspective. The correctness of encryption requires that decryption of an encrypted message under corresponding keys produces the original message. The correctness of E-ART has been proven mathematically and empirically as shown in the following sub-sections.

4.4.1 Empirical Proof

For the empirical test, a 100 documents has been used from i2b2 clinical notes [157] written in an unstructured, informal format. In total in all files, there are 56,054 encrypted

words. All the words were regenerated successfully. The process of encrypting and decrypting E-ART is shown in Figure 4-7. Figure 4-7(a) shows a sample of clinical notes as input data. Figure 4-7(b) shows encrypted data after applying Algorithm 1, and Figure 4-7 (c) shows decrypted data using Algorithm 2.

```
Record date: 2083-08-26
SILVER RIDGE EMERGENCY DEPT VISIT

OROZCO,KYLE    560-40-78-5                        VISIT DATE: 08/26/83

This is a preliminary dictation.

PRESENTING COMPLAINT and HISTORY OF PRESENTING COMPLAINT:  This patient
 is a 58-year-old male with a history of hypertension and diabetes who
presents with fever and chills.  He is four weeks status post revision
of an aorto-bifemoral bypass graft, which got infected and had to be revised.
```

(a)

```
48qn}m}smysy#}qtrtwsn"aR]#xymnqph!r|!rn{n|tlxrz"}sspuwtm+'+""lZWO!P`NW[SL"XO#`!+ww++++4I[_O4SV\#T_U\!
IXLY"a#XWNLSPG#RZ#ZGR\NM]ROWRT!aS\#\Wa_]N[M"JZR#PO\O\TLN!KXNZ#Z\K]O!_S]#^ZYTTO4#!y["XO"ZRMO!K
[[WN#NNaLKN#PRMM#O[LXMYRR#QZ#`T#_SNMS5`Y[[TRNaT#_HRaMO"ZP`ZN5"KYY_X#YSM#WT[[^M]^!_S^!Z_\"LR#^]"N\KWM\
\4"#z[!Y_N#^]\R#^QYSY"J[VT5#^MN!RRI#QP]N[SMM#X[O[#IXNY"a#UQJ5[Oa^\!\[L[O!NR#1115+zz+#\\YO
\]N#z4!:Y]"ON_L\OXXO"U`JYSKU"L]SP]OaMKP["aM#ZQS\"IaM#122#\]ZO]\M#z37"KZWT\"Z]!K`N#M_WWTZ"▯^JXV"\SO#LZ
[#Z]K[O93#!x\"RN\M\SLO!ZRP!]J_UL_NWSR3!"z\#X_M"Ya]!RS"^P_WS_[\#\PQT"NZ[!OXN\5!_R^"Z_N!`\[T!YST[!
QS#xl"r_Z]YVTXS"\RN!MY]#PaMM#MKS!SQSLYM#N\^SR^_OH"NS!YXN#[N_[M#XR\[_LWRT3!57686957
```

(b)

```
Record date: 2083-08-26
SILVER RIDGE EMERGENCY DEPT VISIT

OROZCO,KYLE   560-40-78-5                    VISIT DATE: 08/26/83

This is a preliminary dictation.

PRESENTING COMPLAINT and HISTORY OF PRESENTING COMPLAINT:  This patient
 is a 58-year-old male with a history of hypertension and diabetes who
presents with fever and chills.  He is four weeks status post revision
of an aorto-bifemoral bypass graft, which got infected and had to be revised.
```

(c)

Figure 4-7: Example before and after encryption by E-ART algorithm of a) the original clinical note, b) the encrypted note, and c) the decrypted note

4.4.2. Mathematical Proof

The research proves that the combined process of encrypting and decrypting a message correctly results in the original message. The proof shows that the decryption function basically reverses the encryption steps. The statement of correctness is as follows:

$$\forall\, m, k \quad (E_K(m)) = \text{ENC} \quad D_K\,(ENC) = m$$

Where m ϵ M, k ϵ Z, m is the character to be encrypted, M represents the ASCII table value from 0 to 127, k is the key, Z represents the natural number, and ENC is the encrypted character.

Recall the following:

- $offset_{\ var}$ is generated using *N* as shown in eq. (4.3), and $offset_{dynamic}$ is generated using *Variance* where *N* and Variance ⊆ K as shown in eq. (4.4).
- Every character will be converted to its ASCII value equivalent so $m = Val_{org}$.

There are two cases in the encryption and decryption process. In Case 1, X is greater than Len_{max}. In Case 2, X is less than the Len_{max} as shown in eq. (4.2).

Case 1

Case 1 is when X is less than Len_{max} as shown in eq. (4.2). In this case the *Quotient* = 0 and the encryption algorithm and decryption as follow:

$$E\ (n,k) = X + offset_{dynamic} \tag{4-6}$$

$$D\big(E\ (n,k)\big) = \Big[Len_{max} - \big[\big(Len_{max}\ x\ Quotient\ +\ ENC\ -\ Offset_{dynamic}\big) - Offset_{var} - Offset_{const}\big]\Big] + 1 \tag{4-7}$$

In this case, *Quotient =0* in the decryption algorithm. The following proof shows that that the combined process of encrypting and decrypting a message correctly results in the plaintext.

Proof:

Given that

$X = Val_{Initial\ ref} + offset_{\ var} + offset_{const}$

$Val_{Initial\ ref} = \big(Len_{max} - Val_{org}\big) + 1$

Expanding the value of X from the above definitions

$$E\ (n,k) = \Big(\big(Len_{max} - Val_{org}\big) + 1\Big) + offset_{\ var} + offset_{const} + offset_{dynamic} \tag{4-8}$$

Replacing the value of *ENC* from eq. (4.7) to (4.6)

$$\Rightarrow D\big(E\ (n,k)\big) = \Big[Len_{max} - \big[\big(Len_{max}\ x\ Quotient\ +\ \big(\big(\big(Len_{max} - Val_{org}\big) + 1\big) + offset_{\ var} + offset_{const} + offset_{dynamic}\big) - Offset_{dynamic}\big) - Offset_{var} - Offset_{const}\big]\Big] + 1$$

$$\Rightarrow \quad D\big(E\ (n,k)\big) = \big[Len_{max} - Len_{max} + Val_{org} - 1 - offset_{\ var} - offset_{const} - offset_{dynamic} + Offset_{dynamic} + Offset_{var} - Offset_{const}\ \big] + 1$$

Solving for Val_{org},

$$\Rightarrow \quad \boldsymbol{D(E(n,k)) = Val_{org}}$$

$$\boldsymbol{Val_{org} = m}$$

$$\boldsymbol{D(E(n,k)) = m}$$

Case 2:

Case 2 is when X is greater than Len_{max} as shown in eq. (4.2). In this case, an $offset_{const}$ added to the encryption and decryption process and set *Quotient* = 1 as follow:

$$E(n,k) = X + offset_{const} + offset_{dynamic} \tag{4-9}$$

$$D(E(n,k)) = \left[Len_{max} - \left[\left(Len_{max}\ x\ Quotient + ENC - Offset_{dynamic}\right) - Offset_{var} - Offset_{const}\right]\right] + 1 + offset_{const} \tag{4-10}$$

Expanding the value of X from the previous definitions

$$E(n,k) = \left(\left(\left(Len_{max} - Val_{org}\right) + 1\right) + offset_{var} + offset_{const}\right) mod\ Len_{max}) + offset_{const} + offset_{dynamic} \tag{4-11}$$

Replacing the value of *ENC* from equation (4-12) to (4-9)

$$\Rightarrow D(E(n,k)) = \left[Len_{max} - \left[\left(Len_{max}\ x\ Quotient + (\left(\left(\left(Len_{max} - Val_{org}\right) + 1\right) + offset_{var} + offset_{const}\right)\%Len_{max}) + offset_{const} + offset_{dynamic}\right) - Offset_{dynamic}\right) - Offset_{var} - Offset_{const}\right]\right] + 1 + offset_{const}$$

$$\Rightarrow D(E(n,k)) = \left[Len_{max} - Len_{max} - Len_{max} + Val_{org} - 1 - offset_{var} - offset_{const}\ mod\ Len_{max} - offset_{const} - offset_{dynamic} + offset_{dynamic} + offset_{var} + offset_{const}\right] + 1 + offset_{const}$$

Solving for equal and opposite terms,

$$\Rightarrow D(E(n,k)) = \left[-Len_{max} mod -Len_{max} + Val_{org}\right]$$

Knowing that $\Rightarrow -(Len_{max}\%Len_{max}) = 0$

$\Rightarrow \boldsymbol{D\big(E\ (n,k)\big) = \ Val_{org}}$

$\boldsymbol{Val_{org} = m}$

$\boldsymbol{D\big(E\ (n,k)\big) = m}$

The proof of correctness starts by expanding the definition of $E\ (n,k)$ by replacing the value of X with its definition and then combined $E\ (n,k)$ and $D\big(E\ (n,k)\big)$ by replacing the value of ENC. Solving the equation proof that t the decryption function basically reverses these steps of the encryption.

4.5. Summary

This chapter has addressed the challenge of ensuring transmission and storage security of sensitive data while maintaining low computational and latency overheads. The new lightweight, flexible, and secure encryption algorithm E-ART has been proposed for this purpose. It adopts the dynamic key concept along with a balanced binary search tree data structure and ASCII. The algorithm encrypts the ASCII characters from 32 to 127 which includes all the printable characters plus space.

E-ART's design passed through three stages. In Stage 1, the subtitled method was designed based on the concept of a reflection-balanced binary tree and the ASCII value. In Stage 2, two offsets were added to the algorithm design and tested in order to overcome the issues in Stage 1. In Stage 3, the complexity and randomness were increased by adding a dynamic offset generated from the secret key. Finally, a simple proof methodology has been used to support the functional correctness of the proposed algorithm.

The proposed algorithm uses a single round, which requires few processes and ensures good cryptographic performance. E-ART adopts several operations during the encryption process to achieve a high level of efficiency and security. Those operations include the

concept of balanced tree data structure and ASCII values of text characters to encode data. In addition, it adopts dynamic keys based on a pseudo-random generator where each character in the text document is encrypted with a different cryptographic key. The character's position is used as a seed in the random number generation function to produce the pseudo-random number. This ensures a high level of security against classical and modern powerful attacks. Such security is traditionally guaranteed by increasing the key size without sacrificing performance as is illustrated in Chapter 5.

Chapter 6 uses the outcome of this chapter to design and implement a new de-identification model to de-identify health textual file.

Chapter 5 : E-ART Performance and Security Evaluation

5.1. Introduction

This chapter presents the validation of the proposed E-ART cryptographic algorithm through a series of experiments. An experiment methodology was adopted to evaluate the proposed algorithm.

The performance and security of E-ART were compared to the benchmarked symmetric encryption algorithms AES-128 and DES. Performance was compared in terms of processing time and memory usage. Security was assessed through the avalanche effect, BIC, frequency analysis, and the NIST test.

The chapter is organised as follows: Section 5.2 presents the experimental setup; Section 5.3 describes the evaluation metrics, including the performance and security parameters; Section 5.4 presents the results including performance and security analysis; Section 5.5 presents the security against attacks and Section 5.6 summarises the chapter.

5.2. Experimental Setup

The proposed E-ART algorithm was experimentally evaluated and compared with the widely used encryption algorithms AES and DES. The algorithms were implemented using the Java programming language in console mode to minimize the load from a user interface and other libraries, contributing to more realistic results. All experiments were conducted on the same platform using Windows machine equipped with 32 GB of memory and an Intel i7 3.4 GHz CPU.

NetBeans, an open-source integrated development environment (IDE), was installed to compile and run Java programs. NetBeans enables applications to be developed from a set of modular software components known as *modules*. Each module provides a well-

defined function. Figure 5-1 shows snapshoot of the developed algorithm on NetBeans. The results from the Java programs were then copied to the MATLAB software for graphical representation and comparison.

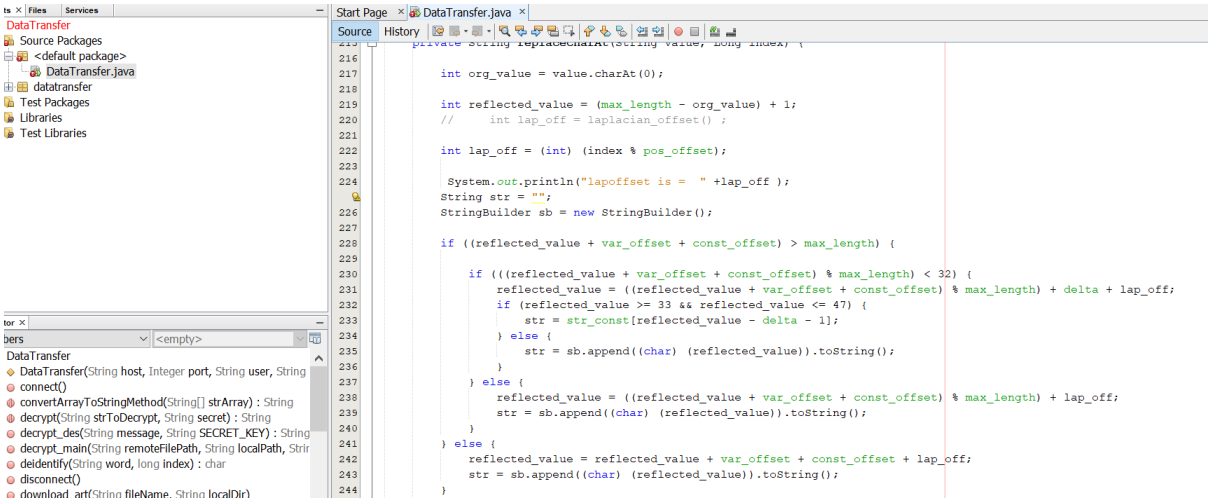


Figure 5-1: Snapshot of Implement Algorithm in NetBeans

5.3. Evaluation Metrics

This section presents the evaluation metrics used to assess E-ART algorithm in term of performance and security as follow:

5.3.1. Performance Parameters

After security, processing time is the most critical criterion for an encryption algorithm, especially in large-sized and real-time applications such as health data [166]. In such applications, heavy processing and long runtimes are undesirable, which is why execution time (encryption/decryption) and memory usage metrics were considered in this performance evaluation. A brief explanation of the parameters considered is given as follows:

1. Memory consumption: The amount of memory consumed during the encryption or decryption processes, measured in megabytes MB. The experiment used getRuntime() and totalMemory() function provided by Java library to calculate memory consumption as explained in Appendix II

2. Encryption and decryption time: The amount of time consumed during the encryption or decryption processes, measured in MS. The experiment used currentTimeMillis() function provided by Java library to calculate the time before and after the encryption.as explained Appendix II
3. File size: The size of the encryption file before and after encryption, measured in KB.

The results for the proposed algorithm were compared with those of two well-known symmetric encryption algorithms, AES-128 and DES. File sizes were used ranging from 200 to 2000 KB. Figure 5-2 illustrates the performance analysis process.

5.3.2. Security Parameters

Security analysis is an essential aspect to consider when assessing the quality of any encryption scheme. To resist well-known statistical attacks, the most basic requirement of a cipher is that the input plaintext and generated ciphertext should be statistically independent. The security of the proposed cipher can be assessed using various security tests, including the following:

- Avalanche effect test: The avalanche effect criterion is met if all output bits change with a probability of 50% when a single input bit is altered [86], as explained in more detail in section 2.4.1.

- Autocorrelation test: This test is met if two output bits change independently of each other when a single input bit change [87]. The weaker the correlation between the input and outputs, the greater the immunity of the algorithm to differential cryptanalysis, as described in section 2.4.1.

- Frequency analysis: To resist the frequency analysis attack, the frequency histogram should be distributed in a relatively uniform way, and it should differ significantly from that of the plaintext [89] as explain in section 2.4.1.

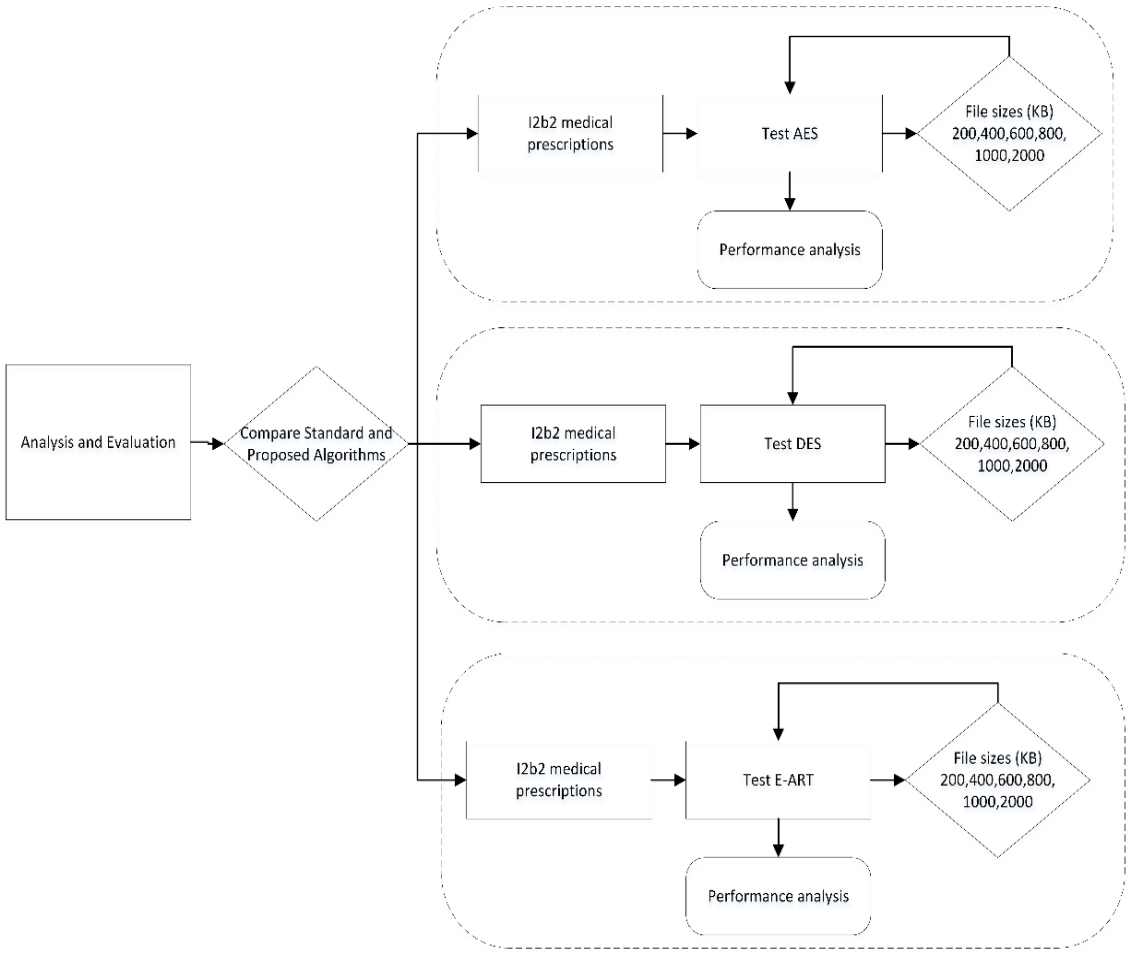


Figure 5-2: Performance Analysis Process

- ***Randomness Verification***

Randomness refers to the degree of difficulty in predicting the next element in a sequence of numbers. It is an essential aspect underlying the security of cryptographic algorithms. Several encryption algorithms use pseudo-random number generators (PRNGs) to construct the encryption key; their security is based on the statistical characteristics of these PRNGs. Thus, suitable metrics are needed to investigate the degree of randomness for the binary sequences produced by PRNGs.

In practice, statistical testing is used to gather evidence that a PRNG indeed produces numbers that appear to be random. These tests address how the observed statistics of the analysed feature fit the expected statistics. There are numerous statistical tests available to measure the randomness of the outputs of a PRNG, including those grouped in the

batteries of NIST [152], Diehard [167], and TestU01[168]. This thesis selected the NIST test because it is one of the most frequently used in the literature and it was published as a NIST standard and used to prepare many formal certifications and approvals. Furthermore, unlike TestU01, which requires integrating the pseudo-random generator into the test suite [168], NIST does not require building the generator into the program; it can simply pip the raw binary output from this pseudo random generator into NIST suit.

The tests focus on the randomness of data according to various statistics relating to bits, including the proportion of bits, frequency of bit change (runs), and cumulative sums. All tests are parameterised by n, which denotes the bit length of a binary sequence to be tested. Table 5-1 summarises suitable values of n for each test recommended by NIST [152].

Table 5-1: Recommended size of Bit-stream for each NIST test

Test Name	Recommended size of n	Subtest
Frequency	$n \geq 100$	1
Frequency within a block	$n \geq 100$	1
Runs	$n \geq 100$	1
Longest run of ones	$n \geq 1128$	1
Rank	$n > 38\ 912$	1
Spectral	$n > 1000$	1
Non-overlapping T. M.	$n \geq 8m - 8$	148
Overlapping T.M.	$n \geq 106$	1
Maurer's universal	$n > 387\ 840$	1
Linear complexity	$n \geq 10^6$	1
Serial		2

Approximate entropy		1
Cumulative sums	$n \geq 100$	21
Random excursions	$n \geq 10^6$	8
Random excursions variant	$n \geq 10^6$	18

5.4. Results

This section presents the result of the performance and security analysis conducted on E-ART algorithm and compared it with AES and DES algorithms. The Performance analysis including processing time, memory usage and the size of the ciphertext file. Security was assessed through the avalanche effect, BIC, frequency analysis, and the NIST test.

5.4.1. Performance Analysis

To maintain high performance during encryption and decryption of large datasets, the parameters of memory consumption, time consumed, and file size were tested and measured. The results in Table 5-2: AES Performance, Table 5-3: DES Performance and Table 5-4: E-ART Performance indicate that the proposed algorithm outperformed AES and DES in terms of processing time, with comparable memory usage for file sizes between 200 and 1000 KB, and slightly higher memory usage for 2000 KB.

Table 5-2: AES Performance

File size (KB)	Encryption		Decryption	
	Processing Time (MS)	**Memory (MB)**	**Processing Time (MS)**	**Memory (MB)**
200	1433	17	1378	17
400	1956	21	1363	18
600	2118	27	1637	25
800	2335	30	1645	31
1000	2528	33	1995	35
2000	3616	55	1754	56
Average	2331	30	1628	30

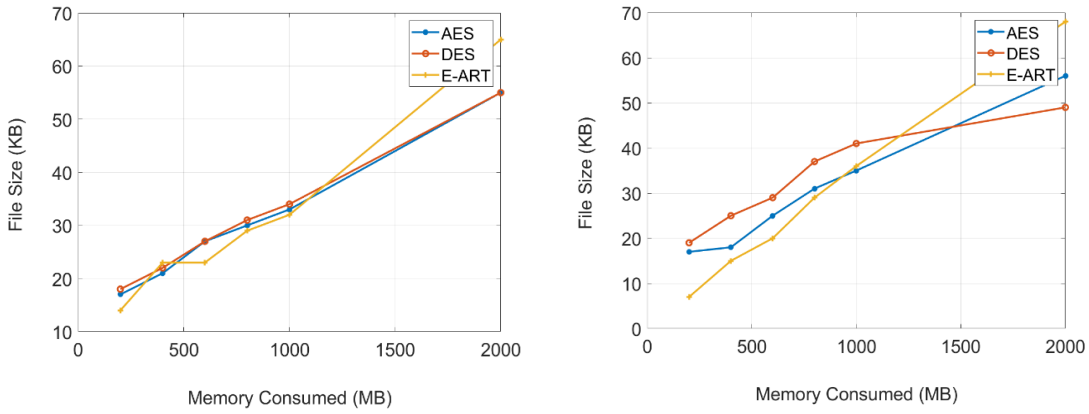
Table 5-3: DES Performance

File Size (KB)	Encryption		Decryption	
	Processing Time (MS)	Memory (MB)	Processing Time (MS)	Memory (MB)
200	1838	18	1992	19
400	2067	22	2444	25
600	2190	27	2750	29
800	2575	31	3183	37
1000	3034	34	3658	41
2000	4537	55	5500	49
Average	2706	31	3254	33

Table 5-4: E-ART Performance

File Size (KB)	Encryption		Decryption	
	Processing Time (MS)	Memory (MB)	Processing Time (MS)	Memory (MB)
200	123	14	162	7
400	189	23	250	15
600	253	23	320	20
800	413	29	385	29
1000	479	32	460	36
2000	1854	65	775	68
Average	551	31	392	29

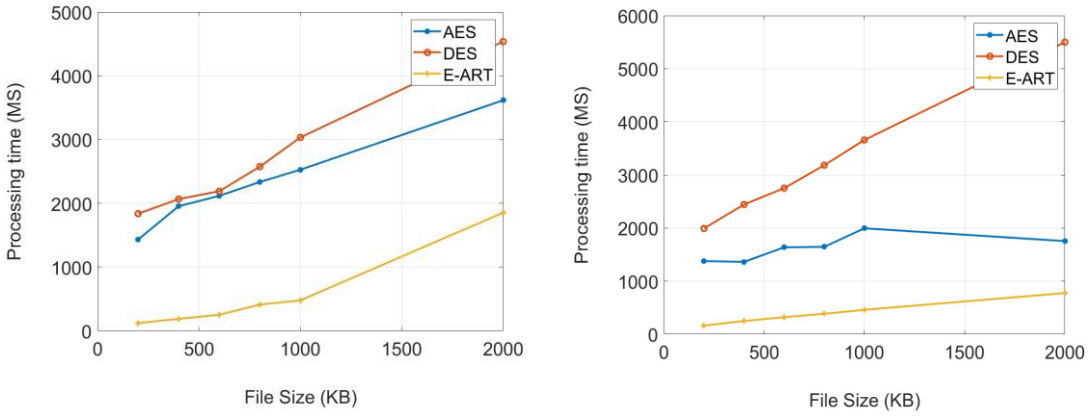
Figure 5-3 illustrate the memory consumed by encryption and decryption in E-ART compared to AES and DES for different files size. As the figures indicate, all three algorithms consumed nearly the same amount of memory for small file sizes ranging from 200 KB to 1000 KB, while E-ART consumed slightly higher for file sizes greater than 200 KB.



(a) **(b)**

Figure 5-3: Comparison of Memory Consumption of (a) Encryption (b) Decryption for AES, DES and E-ART Algorithm

Figure 5-4 shows a comparison of the execution time to encrypt and decrypt different-sized text files for E-ART compared to AES and DES. E-ART had the shortest running time with the highest level of 1854 MS for file sizes of 2000 KB compared to 4537 MS and 3616 MS for DES and AES, respectively. Therefore, it can be concluded that the implementation of the reflection of the balanced tree, along with the dynamic offset, achieves high computational speed with minimal memory usage.



(a) **(b)**

Figure 5-4: Comparison of execution time of (a) Encryption and (b) Decryption for AES, DES, and E-ART algorithms

Table 5-5: Comparison of file size before and after encryption for AES, DES, and E-ART algorithms shows the file size before and after encryption for each encryption algorithm. File size increased by approximately 31% compared to the plain-text file under the AES and DES algorithms, but it increased only by around 5% for files encrypted using E-ART. This illustrates that using E-ART does not have as large an influence on file size compared to AES and DES. Hence, it is suitable for encrypting large amounts of data and thus holds promise in the domain of data encryption strategies for big data.

Table 5-5: Comparison of file size before and after encryption for AES, DES, and E-ART algorithms

Plain-text File Size (KB)	File Size after Encryption		
	AES	**DES**	**E-ART**
200	264	264	211
400	527	527	422
600	790	790	633
800	1053	1053	843
1000	1316	1316	1054
2000	2632	2632	2107
Average increase (%)	˜ 31%	˜ 31%	˜ 5%

5.4.2. Security Analysis

In this section, the proposed algorithm is evaluated and tested against popular crypto-analysis and statistical attacks.

Avalanche Effect

The avalanche effect is measured using the Hamming distance. The Hamming distance between two texts of equal length refers to the number of positions at which corresponding characters are different. To measure the avalanche effect, the research used 10 pair keys that differed only in one bit to encrypt 10 pair files with each encryption technique. In turn, the Hamming distance is calculated to obtain the number of bits that differed between each pair of files. The avalanche effect is calculated as

$$AvalancheEffect = \frac{HammingDistance}{TotalNoCharacters} \times 100 \quad (5\text{-}1)$$

The experiment used python script that convert the files to binary form then compare each avalanche pairs in bit level to calculate the hamming distance and the avalanche effect as shown in Appendix II. Table 5-6 shows the average Hamming distance and the avalanche effect for 10 pairs of files encrypted using E-ART, AES, and DES. The average avalanche effect of the proposed algorithm was 50.11% compared to 49.2% for AES and 49.3% for DES. These results indicate that the proposed encryption system satisfies the avalanche effect criterion and thus provides strong protection against differential cryptanalysis.

Table 5-6: Avalanche Effect Performance

Technique	Hamming distance	Avalanche effect
E-ART	57852	50.1%
AES	39345	49.2%
DES	39425	49.3%

Bit independent criteria

To measure the degree of independence between the pair's j and k of avalanche variables, the avalanche variables were first created by changing a single bit in the input key. In turn, the avalanche variables were converted to binary representation. Following this, the correlation coefficient between j and k was calculated. Table 5-7: BIC Performance shows a comparison of the BIC result for the proposed algorithm, AES, and DES. E-ART yielded a correlation value of 0.1863 compared to 0.1818 for AES and 0.1847 for DES.

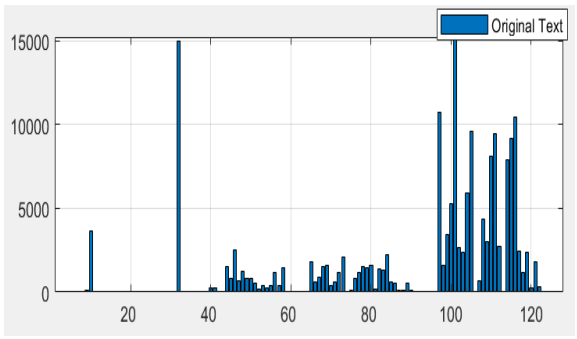
Table 5-7: BIC Performance

Technique	Bit Independence
E-ART	0.1863
AES	0.1818
DES	0.1847

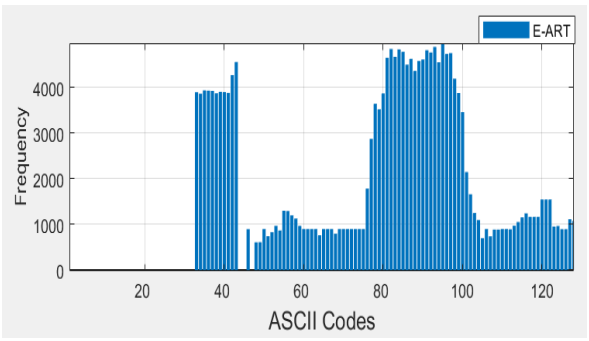
The results indicate very low dependence between the two avalanche variables. Hence, the proposed algorithm satisfies the BIC, which means that it is difficult to predict one bit based on the other bits, thereby increasing the difficulty of cryptanalysis.

Frequency Analysis

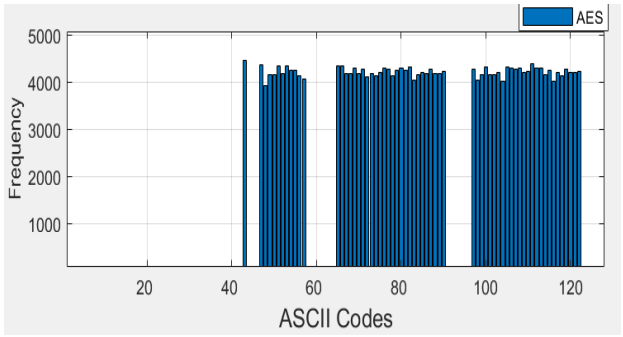
The generated histograms of the plain-text file that contains 31513 words and its corresponding cipher-text are shown in Figure 5-5. The histogram of the encrypted text is distributed in a relatively uniform way and differs significantly from that of the plain-text



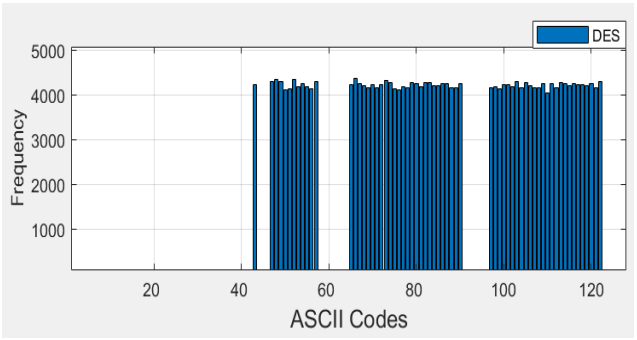
(a)



(b)



(c)



(d)

Figure 5-5: Histograms of (a) plain-text, (b) E-ART, (c) AES, and (d) DES

Table 5-8: ASCII Histogram Frequency Value

Character value (Decimal)	Equivalent character	Frequency (Plain-text) (%)	Frequency (AES) (%)	Frequency (DES)(%)	Frequency (E-ART)(%)
40	(	0.136	0	0	1.805
45	-	0.409	0	0	0
50	2	0.409	1.547	1.523	0.416
55	7	0.1820	1.577	1.551	0.602
60	<	0	0	0	0.416
65	A	0.910	1.615	1.566	0.416
70	F	0.182	1.551	1.562	0.416
75	K	0.068	1.559	1.529	0.416
80	P	0.796	1.599	1.575	1.791
85	U	0.295	1.559	1.553	2.214
90	Z	0.045	1.571	1.572	2.136
95	_	0	0	0	2.304
100	d	2.594	1.602	1.565	1.601
105	i	4.756	1.602	1.579	0.323
110	n	4.005	1.570	1.497	0.416
115	s	4.528	1.580	1.557	0.534
120	x	0.136	1.565	1.572	0.716
125	}	0	0	0	0.415

Table 5-8 describes and Figure 5-5: Histograms of (a) plain-text, (b) E-ART, (c) AES, and (d) DES(a-d) illustrates the values in the histograms. It shows the frequency for some characters with their decimal value for the plain-text and cipher-text, which was encrypted by AES, DES, and E-ART. Frequency refers to the average number of times that a given

character is repeated in the text. Table 5-8 indicates that good confusion was achieved for all of the tested encryption algorithms. For example, the character value 105 (105 Dec = i character) has a frequency of 4.75% in the plain-text while it has a frequency of 0.323% in the E-ART cipher-text. Also, the character value 90 (90 Dec = Z character) has 0.045 % frequency in the plain-text with 2.136% frequency in the E-ART cipher-text. As such, almost all of the frequency values are significantly different compared to the plain-text, which will confuse the adversary and secure the cipher-text against cryptanalysis.

Randomness Analysis: NIST Test

NIST Statistical Test Suite [152] is a statistical package consisting of 15 tests designed to assess the degree of randomness for binary sequences produced by cryptographic random number generators as shown in Figure 5-6. These tests compute the *P*-value to determine the strength of the evidence against the null hypothesis. In each test, the *P*-value is the probability that an ideal random number generator generates a sequence that is less arbitrary than the series tested according to the types of non-randomness evaluated by the test. The significance level α, which typically ranges from 0.001 to 0.01, can be set at 0.01. If $P \geq \alpha$, then the null hypothesis is accepted and the sequence appears to be random.

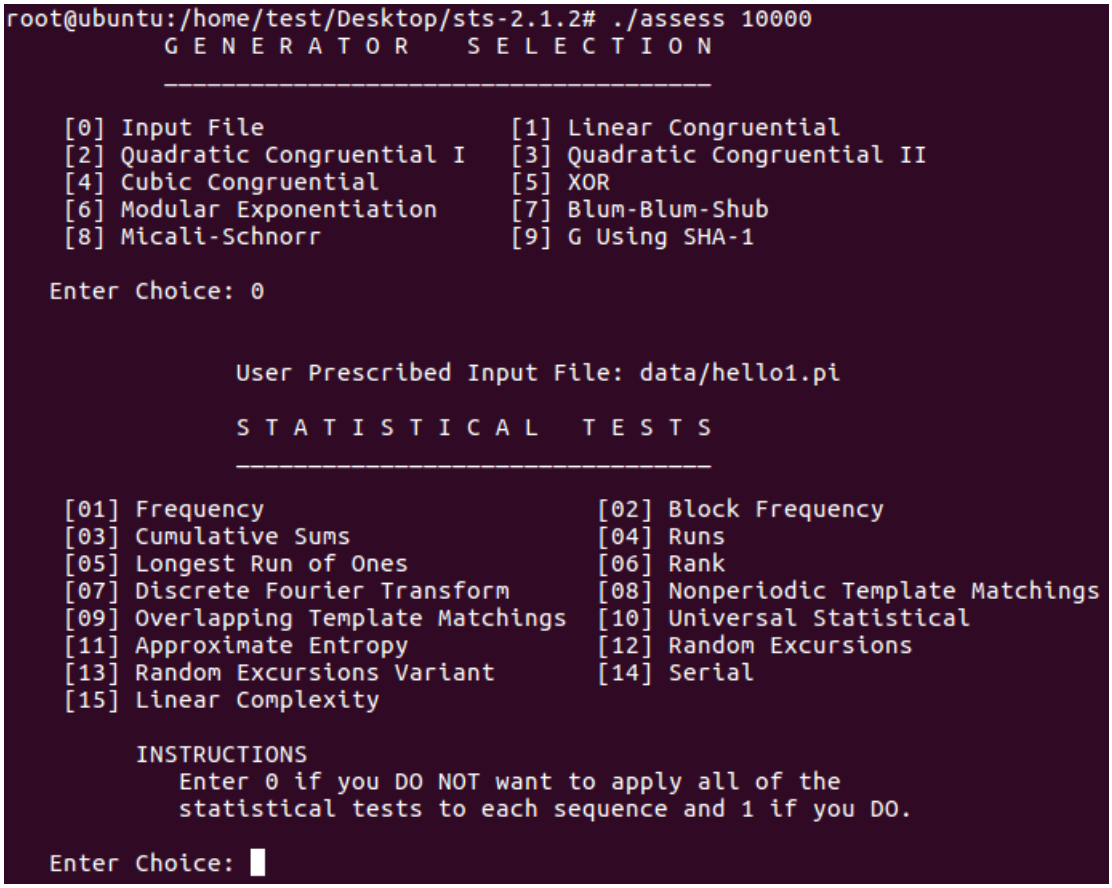


Figure 5-6: Snapshot of NIST Test

To evaluate the quality of the dynamic offset, the research used NIST statistical tests [152] with 1000 sequences, where each series was 128 bits. For this research's experiment, selected five tests to determine the randomness and non-uniformity of the dynamic key. The selection was based on the input size recommendation given by the NIST user guide [152], which is explained in Table 5-1, as the key restricted to 128 bits. In the next sections, a brief explanation is given of the five tests that are applicable to the proposed key length.

- **Frequency (Monobit) Test**

  The test aims to calculate the number of zeros and ones in the sequence. If the number of zeros is almost the same as the number of ones in the entire sequence, then the sequence is considered truly random.

- **Frequency Test within a Block**

  The test aims to check if the frequency of ones within an M-bit block is approximately M/2, as expected under the randomness assumption.

- **Runs Test**

  The focus of this test is to assess the expected total number of runs within the sequence, where a run is a consecutive sequence of identical bits. It aims to determine whether the fluctuation between such zeros and ones is too fast or too slow.

- **Cumulative Sums Forward Test**

  The test aims to determine whether the cumulative sums at the early stage of the measured sequence is too large or too small in contrast to the expected behaviour of that cumulative sum for random sequences.

- **Cumulative Sums Backward Test**

  The test aims to determine whether the cumulative sums at the early stage of the measured sequence is too large or too small in contrast to the expected behaviour of that cumulative sum for random sequences.

Using the proposed pseudorandom generator as mentioned on section 4.1.3, 1000 sequences have been produced. The results of the five statistical tests are presented in Table 5-9. The *P*-value of ≥ 0.01 in each test, as shown in Table 5-9, indicates that the proposed algorithm generates dynamic offsets that are truly random and infeasible to predict.

Table 5-9: Statistical Test Results

Test	P-value	Remarks
Frequency test	0.7399	Random
Block frequency test	0.7399	Random
Runs test	0.0668	Random
Cumulative sums forward test	0.1223	Random
Cumulative sums backward test	0.5341	Random

5.5. Security against Attacks

The experiments were undertaken to prove that the proposed cipher is secure and efficient. In this section, a brief discussion of cryptanalysis is given to validate the security of the proposed algorithm and to prove that it can resist various well-known attacks.

5.5.1. Brute Force Attack

To resist brute-force attacks, the key space should be sufficiently large. Alvarez and Li [169] stated that for an encryption algorithm to resist brute-force attacks, its key space must be at least 2^{100}. The attacker rely on factors such as key length and calculation speed [170]. One way to calculate the brute-force attack time [171] [170] is as follows:

$$\boldsymbol{Estimated\ speed} = \frac{\boldsymbol{2^{key\ lenth}}}{\boldsymbol{speed}} \quad (5\text{-}2)$$

In the E-ART algorithm, the initial key can be set to be 64 or 128 bits. This makes the search space for the initial key with 64 bits involve 2^{64} possible solutions, and for 128 bits, the number of solutions is 2^{128} .

Table 5-10 displays the estimated time to find the key under a brute-force attack. Considering a typical desktop computer with a speed of 2,000,000,000 Hz, equation (5-2) can be used to estimate the time needed to search the key. To resist brute-force attacks, the encryption system must have at minimum 2^{100} key pools [169]. This is achieved if the key

is set to 128 bits, but not with a 64-bit key. However, Section 4.2.3 indicates that the E-ART relies on the three parameters of R, Pseudo, and *Variance* to generate dynamic offsets of 128 bits, which makes the key pool 2^{192} in total (in the case of an initial key of 64 bits) or 2^{256} in total (in the case of an initial key of 128 bits), which is a wide range. Thus, the cipher is computationally secure against brute-force attacks.

Table 5-10: Time Complexity for Brute-force Attack

Key space	Possible combinations	Approximate time (in seconds)
$\mathbf{2^{64}}$	18,446,744,073,709,551,616	9,223,372,036.854775808
$\mathbf{2^{128}}$	3.402823669209384634633746074317e+38	170,141,183,460,469,231,731,687,303,715.89
$\mathbf{2^{192}}$	6.277101735386680763835789423207e+57	3.138550867693340381917894711603e+48
$\mathbf{2^{256}}$	1.157920892373161954235709850086e+77	5.789604461865809771178549250434e+67

5.5.2. Chosen and Known Plain-text/Cipher-text Attacks

The proposed cipher includes a dynamic key structure that varies for each character. Therefore, no critical information can be captured from the collected encrypted messages since they are encrypted with different dynamic keys. The obtained cipher-text is also significantly different from the original message, as shown in Figure 5-5: Histograms of (a) plain-text, (b) E-ART, (c) AES, and (d) DES. Thus, the adversary cannot establish any relationships among received encrypted messages**.** Consequently, the proposed scheme can be considered secure against chosen and known plain-text/cipher-text attacks.

5.5.3. Statistical Attacks

In addition to the high random outcomes of the pseudorandom generator, as measured by the NIST test, a high degree of randomness is ensured based on a BIC test by having a

correlation coefficient that is close to zero. Furthermore, the key sensitivity to any bit of the secret key is achieved according to an avalanche effect test. The frequency values of the cipher-text are significantly different compared to those of the plain-text, which will confuse adversaries and secure the cipher-text against cryptanalysis. Consequently, the proposed scheme can be considered secure against statistical attacks.

5.6. Summary

Existing symmetric encryption algorithms depend on static keys and multi-round functions to reach the required security levels. This entails a trade-off between security and performance. The objective of this thesis was to propose a simple and efficient algorithm that reduces the required latency and resources without compromising security. For this purpose, the E-ART algorithm was proposed in Chapter 4 using the concept of the reflection of a balanced binary tree and the ASCII representation.

This chapter focused on addressing the following research question:

- Can the proposed encryption algorithm achieve a trade-off solution between security and performance compared with the standard AES and DES?

To answer this question, an empirical evaluation of the proposed algorithm based on a series of experiments was presented. The performance of the E-ART algorithm was compared to the widely used encryption algorithms, AES and DSS, in terms of processing time and memory usage. Security was assessed through the avalanche effect, frequency analysis, and the Randomness test.

In terms of performance, the results indicate that E-ART had the shortest running time and comparable memory usage for file sizes between 200 and 1000 KB, and slightly higher for 2000 KB (see Table 5-2: AES Performance Table 5-3: DES Performance, Table 5-4: E-ART Performance . Table 5-5: Comparison of file size before and after encryption for AES, DES, and E-ART algorithms shows that ciphertext increased by only 5% compared

to the plain-text file under E-ART, while it increased by around 31% for files encrypted using AES and DES. This makes E-ART a promising solution for big data, delay-sensitive, and real-time applications.

For security analysis, the avalanche effect analysis showed that E-ART changed half of the bits on average in the cipher-text when a single bit was changed in the initial keys. This demonstrates that the algorithm is sufficiently sensitive to any change in the key and satisfies the avalanche effect criterion, making key-related attacks considerably more difficult to perform successfully. The BIC analysis performed on two avalanche variables showed a satisfactory bit independence criterion (i.e., the obtained correlation value was close to 0 as shown on Table 5-7). Therefore, it is difficult to predict one bit from the other bits, which heightens the difficulty of cryptanalysis. Furthermore, the size of the initial key can be set to 64 or 128 bits, whereas the size of the proposed pseudorandom can be set to 64 or 128 bits. These sizes are sufficiently large to make brute force attacks unfeasible. The histogram of the encrypted text is relatively uniformly distributed and differs significantly from that of the plain-text, as shown in Figure 5-5: Histograms of (a) plain-text, (b) E-ART, (c) AES, and (d) DES..

The performed NIST tests showed that the dynamic offset has a satisfactory level of randomness and uniformity. Given that the dynamic offset changes for every character, it is also considerably more difficult to perform algebraic [172], linear, and differential attacks [173] successfully. Overall, the results show that the proposed cipher is a good candidate for lightweight modern text data encryption.

Chapter 6 : ARTPHIL De-Identification Model Design and Evaluation

6.1. Introduction

Privacy-preserving of data has become a significant issue, while the applications and capabilities of big data are expanding dramatically fast. There is no doubt that this expansion creates enormous opportunities and avenues to understand and solve big problems over varying domains. However, the privacy and security concerns about Big Data are also growing. Legal systems establish laws to protect the privacy of individual information disclosed for secondary use. A typical example is the GDPR [46] which outlines a specific set of rules for sharing and storing personal data to protect individual privacy. It states that individuals' explicit consent is required before sharing their data for secondary purposes [77]. However, obtaining explicit consent can be difficult or impractical in some scenarios (e.g. research, Big Data analytics and machine learning). Data minimization also is one of the fundamental principles of GDPR and has strict data retention policies. This implies that personal data can be kept for no longer than necessary to carry out the purpose of processing the data [1]. GDPR points to the pseudonymization as a protective measure for processing personal without obtaining data subject consent or retaining data for a longer period, such as for conducting scientific or statistical research. However, for this exemption to apply, pseudonymization should meet the GDPR standard, and the existing pseudonymization techniques were developed long before GDPR requirements were established. Many implementations of pseudonymization approaches use static pseudonyms for data subjects [143], while others may contain indirect identifiers; in both cases, these fail to protect against re-identification due to privacy breaches arising from linkage attacks.

This chapter is organized as follows: in section 6.2, the problem formulation was stated. In section 6.3, the de-identification of protected health information was briefly presented. In section 6.4, the use of named entity recognition in the de-identification system was described. Section 6.5 presents the state-of-the-art solution. A formulation of the proposed de-identification approach for unstructured data in section 6.6. The experiment that was used for evaluation was presented and discussed in section 6.7. The result was discussed 6.8 and finally, the chapter was summarised in section 6.9.

6.2. Problem Formulation

As discussed in the chapter 2, pseudonymization of data suggested by GDPR [2] as one of the protective measures that controllers can use to "evaluate the feasibility of further processing of personal information for archiving purposes in the public interest, scientific or historical research purposes, or statistical purposes by processing data that do not enable or no longer enable the identification of data subjects". However, GDPR restricts a data handler's potential to benefit from Pseudonymized data if re-identification processes are "reasonably likely to be employed, such as singling out, either by the controller or by another person to identify the natural person directly or indirectly" [47]. Hence, data controllers should implement several technical and organisational measures to ensure that pseudonymous data is disconnected from the key enabling re-identification. Furthermore, the risks of re-identification are dynamic and evolve over time, and this implies that data controllers should evaluate these risks on a regular basis and take necessary action when they become significant. For example, changing pseudonyms over time for each use or each type of use as a way to reduce the risk of re-identification through linkability [143].

In other words, the GDPR provides several regulatory incentives to adopt pseudonymization. There are, therefore, significant benefits associated with using it, which include enabling data processing for secondary purposes without the need to obtain the explicit consent of data subjects. However, for this exemption to apply,

pseudonymization should meet the GDPR standard, and the existing pseudonymization techniques were developed long before GDPR requirements were established [143]. Many implementations of pseudonymization approaches use static pseudonyms for data subjects, while others may contain indirect identifiers; in both cases, these fail to protect against re-identification due to privacy breaches arising from linkage attacks [174][175].

Moreover, previous works in the field of data de-identification including anonymization and pseudonymization have mainly focused on highly-structured data, such as group-based methods ($k - anonymity$ and $l - diversity$); however, these methods are inefficient for textual data with highly variable structure [43]. Even with the availability of de-identification techniques for unstructured data [176]–[179], these techniques tend to remove or generalize sensitive attributes, which makes it difficult to reproduce the original data if needed and authenticate. Encryption is also considered as an efficient method to obtain reversible anonymization and allow later request access to the research data [10]. However, the cumbersome key management and distribution of symmetric encryption algorithm prevent a suitable level of scalability from being provided. In addition, more lightweight and practical alternatives need to be developed [180].

As a result, there is a need for a new reversible model for de-identifying unstructured data in order to address the conflicting requirements of preserving privacy while supporting the legitimate use of data. This chapter addresses this need by integrating Philter [149], the information extracting tool, with the lightweight encryption algorithm EART proposed in chapter 4. The chapter main contributions are as follow

- The proposal of reversible, fast de-identification model ARTPHIL to de-identify unstructured textual health data cost-effectively without compromising security.

- The implementation and evaluation of the proposed model using 2014 i2b2 testing data that annotated to comply with the HIPAA guidelines.

ARTPHIL detects all personal data specified by HIPAA guidelines which include direct and indirect identifiers. Therefore, reduces the risk of re-identification via linkage attacks and helps to comply with GDPR requirements for the lawful processing of personal data.

6.3. De-identification of Protected Health Information under HIPAA

Data security and privacy issues become even more critical when used in a healthcare environment, which typically deals with patient sensitive information. The Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy Rule [3] establish two approaches for de-identifying Protected Health Information: The Expert Determination Method and the Safe Harbor method. Neither method guarantees a reliable de-identification method with no risk of re-identification. Instead, the methods are designed to be practical approaches for creating and sharing de-identified healthcare information with a minimum risk of re-identification [26].

6.3.1. The HIPAA Expert Determination Method

The Expert Determination process requires an expert examining the data and determining an appropriate method of de-identification that minimizes the risk of re-identification. It states that the expert must know and use "generally accepted statistical and scientific principles and methods," [3] This would necessitate familiarity with the relevant literature on statistical disclosure control and de-identification methods. There are strong precedents for de-identification and risk levels that should be used in a suitable de-identification approach.

6.3.2. The HIPAA Safe Harbor Method

The Safe Harbor method specifies 18 data elements [3] that consider identifiable health information. These elements need to be removed or generalized from data to consider anonymized or de-identified. "The 18 types are:

1. Names
2. All geographic subdivisions smaller than a state, including street address, city, county, precinct, ZIP code,
3. All elements of dates (except year) for dates that are directly related to an individual, including birth date, admission date, discharge date, death date, and all ages over 89 and all elements of dates (including year) indicative of such age, except that such ages and elements may be aggregated into a single category of age 90 or older
4. Telephone numbers
5. Fax numbers
6. Email addresses
7. Social security numbers
8. Medical record numbers
9. Health plan beneficiary numbers
10. Account numbers
11. Certificate/license numbers
12. Vehicle identifiers and serial numbers, including license plate numbers
13. Device identifiers and serial numbers
14. Web Universal Resource Locators (URLs)
15. Internet Protocol (IP) addresses
16. Biometric identifiers, including finger and voiceprints
17. Full-face photographs and any comparable images

18. Any other characteristic that could uniquely identify the individual”

6.4. De-identification as Named Entity Recognition

The de-identification of structured data is widely studied in the recent past, and there are variety of techniques for this purpose [12],[181], [182]. However, de-identifying unstructured data, mainly text data, is difficult and requires researchers' manual intervention. The main challenge in de-identifying unstructured data is finding the sensitive attributes that spread throughout the text document.

Several techniques have been proposed to extract the sensitive attributes of unstructured data. Most of them can be seen as an application of Named Entity Recognition (NER). NER is a technique that finds and categorize important words inside the text [183]. Many different natural language processing (NLP) applications are taking benefit of the NER technique. These applications include questioning answering applications [184], [185] tweet analysis [186], [187] automatic text extraction [181], [182] and data mining applications [188].

In de-identification techniques, NER considers a useful tool to extract identifying and sensitive attributes from unstructured text. NER techniques can be classified into five categories which are lexicon-based, rule-based, machine learning, deep learning and hybrid techniques [183]. A brief description of each of the method is provided as follows:

- **Lexicon-based:** This technique is based on a pre-defined list of keywords that match explicit NER to extract the keyword, a matching algorithm is used to compare words by words in the dictionary [189], [190]. A major limitation of this approach is to keep an up-to-date list of keywords in the dictionary that could be a challenging task. Also, a word having multiple synonyms cannot be detected even it matches with the target keyword in the dictionary. To overcome this limitation, a method based on augmented search and replace technique is proposed in [191]. This technique used a dictionary of 1.8 million

names and a list of clinical and common usage words. Similarly, Fuzzy gezzatr approach is adopted to find mistyped instance and stemmed recognition approach to match the stem of each word, not only the actual word.

- **Rule-based technique** consists of a set of rules such as pattern matching approach, hand-crafted rules, heuristics and grammatical rules to recognize NER from unstructured text. A rule-based method using heuristics and grammatical rules is proposed in [192]. This technique is applied to drug-related crime news documents and able to detect type, price and amount of the drugs and nationality of the suspect with 87% accuracy. Rule-based systems' advantages are that they require little or no annotated training data, and that they can be easily modified or updated to improve performance [190]. It also considers an effective method for detecting complex name entity structures that are difficult to detected using other learning models. However, it is quite expensive, domain-specific and lacks the ability of robustness and portability [193].
- **Machine Learning:** these techniques can be categorized into two categories that are supervised learning-based techniques and conditional random fields-based methods. Supervised techniques mainly consist of a support vector machine (SVM) classifier. The results obtained through supervised machine learning techniques are promising. However, these techniques need annotated data for training. Creating annotated data is an expensive task as it requires a lot of time and effort with domain experts' support. Because annotated data is not always available, it is more difficult to use supervised machine learning techniques in NER applications than unsupervised ones.
- **Hybrid model:** this approach combines lexicon and rule-based approaches to take the benefits of both approaches and overcome the limitations of both approaches [12]. A techniques called SPOT is proposed in [194] based on a hybrid model. SPOT is used to identify drug names from large clinical corpora and add it as new dictionary entries to keep the dictionary up to date. It also suggests a phrase that was misplaying. HMS Scrubber [195] is a good example

of combined rule-based and pattern matching methods. It uses 50 regular expressions to detect PHI and replace it with tags indicating its category.

- **Deep learning:** Deep Learning algorithms present a new framework that overcomes drawbacks in earlier methods. These models take vector representations for the inputs (i.e., word embedding) and use multiple ANN layers to automatically learn the relevant features. Several techniques have been proposed in the field, mainly using recurrent neural networks (RNNs) and Long Short-Term Memory networks (LSTMs) [196]. Traditional RNN uses a loop to allow information to persist. It can be thought of as multiple copies of the same network, with each network passing a message to a successor. This chain-like nature of RNN leads to short-term memory issues - the longer the chain is, the more probable that the information is lost along the chain [197]. LSTM overcomes the short-term memory problems by using special units that selectively remember or forget information. Therefore it enables better preservation of long-range dependencies [198]. However, both RNN and LSTM suffer from the problem of uni-directionality. These models only preserve the information from the inputs that have already been passed through the models. This uni-directionality can hinder the model's ability to learn the exact meaning of a word since the model cannot incorporate later parts of a sentence into context. Several papers introduced Bi-directional Long Short-Term Memory Network (Bi-LSTM) to solve this uni-directionality problem [179]. Bi-LSTM takes the inputs and runs LSTM in two ways - forward from left to right and backward from right to left. The model then concatenates the two results. However, such representations cannot take advantage of both the left and right context simultaneously [179].

Table 6-1 compares the three popular PHI detection approaches regarding different requirements such as crafted knowledge, annotated training data, and the need to know all data variety. The major drawback of the rule-based method is that it necessitates the use of qualified experts to encode each rule manually. However, it does not require a large amount of annotated data like the machine learning and deep learning approach

Table 6-1: Comparison of methods used for PHI Detection

Characteristics	Rule-based	Machine learning	Deep learning
Manually crafted knowledge	Required	Not required	Not required
Manually annotated training data	Required	Large amount	Required
Need to know all data variety	Required	Not required	Not Required
Modification	Simple	Difficult	Difficult
Examples	Regular expressions, dictionaries, rules	Support vector machines, conditional random fields, decision trees, Naïve Bayes	recurrent neural networks, Long Short-Term Memory networks
Types of PHI commonly detected	Dates, ages, phone numbers	Addresses, names	Age, data, contact

6.4.1. Evaluation Matrices

For NER, the different evaluation metrics are precision, recall and their harmonic mean, i.e. the F-measure. The F-measure was created to make comparing and contrasting different systems easier. Comparing numerous systems with a single value is easier than comparing them with two values [199]

Precision, also called positive predictive value, is the fraction of the relevant tokens (correctly classified) among the retrieved tokens. recall, also called sensitivity, is the fraction of relevant tokens (correctly classified) that were retrieved. The output of a classifier can be presented in a confusion matrix that shows the number of true-positive annotations (TP), true-negatives annotations (TN), false-positive annotations (FP), and the number of false-negative annotations (FN). Precision eq. 6.2 and recall eq. 6.1 can be computed from such a matrix. F-measure eq. 6.3 is the weighted mean of precision and recall. Recall answers the question; "Did we find all that we were looking for?" and precision answers the question; "Did we only label what we were looking for?". A high recall is generally preferred over high precision, as it measures the percentage of PHI that is correctly identified, and the privacy of the data subjects

is prioritized over a potential loss of document interpretability. To emphasize the sensitivity, F2 measure is calculated in addition to the F1 measure, which weighs recall (twice) higher than precision, as defined in eq. 6.4:

$$Recall = \frac{TP}{TP+FN} = \frac{\#\ of\ corrected\ entites}{\#\ of\ corrected\ prediction} \quad (6\text{-}1)$$

$$Precision = \frac{TP}{TP+FP} = \frac{\#\ of\ corrected\ entites}{\#\ of\ expected\ prediction} \quad (6\text{-}2)$$

$$F_{\ 1} = \frac{2\times Precision\times Recall}{Precision+Recall} \quad (6\text{-}3)$$

$$F_2 = \frac{(\beta^2+1)\times Precision\times Recal}{(\beta^2\times Precision+Recall}\ ,\ \beta = 2 \quad (6\text{-}4)$$

6.5. Stat-of-the-art Solutions

This section presents the current best performance model for data de-identification as follow; 1) Philter model [149] which based on rule-based and statistical models. 2) Dernoncourt et al model [200] based on based on Artificial Neural Networks (ANNs). 3) Liu et al. model [201] based on Conditional Random Field (CRF) and ANNs. Table 6-2 reports the performance of the three models trained on the i2b2 2014 training corpus and evaluated on the i2b2 2014 test corpus.

Table 6-2: Performance of Current State-of-the-art Models

Model	Recall	Precision	F1
Philter model [149]	99.92	78.58	94.77
Dernoncourt et al model [200]	97.83	97.92	97.88
Liu et al. model [201]	97.28	99.30	98.28

6.5.1. Philter

An example of recent text identification application Philter presented in [149] is a customizable open-source de-identification software was developed and evaluated with an extensive collection of unstructured clinical note from UCSF and 2014 i2b2. The algorithm is based on rule-based and statistical natural language processing (NLP) approaches. To identify PHI in free-text documents, the algorithm utilizes an overlapping pipeline of methods that are state-of-the-art in each application, including regular expressions, statistical modelling, blacklists, and whitelists, as shown in Figure 6-1. The entity identified as PHI will be replaced with an obfuscated string of precisely the same length (e.g., "David Smith" becomes "**** *****").

The algorithm compared to the two strongest real-world competitors [149], Physionet [40] and Scrubber [202], on the basis of recall. Philter demonstrated the highest overall recall on both corpora and had the highest recall in each PHI category on both corpora. A major drawback of this approach is that this is irreversible. The removed PHI data cannot be re-generated if an authenticated user later needs full access to the data.

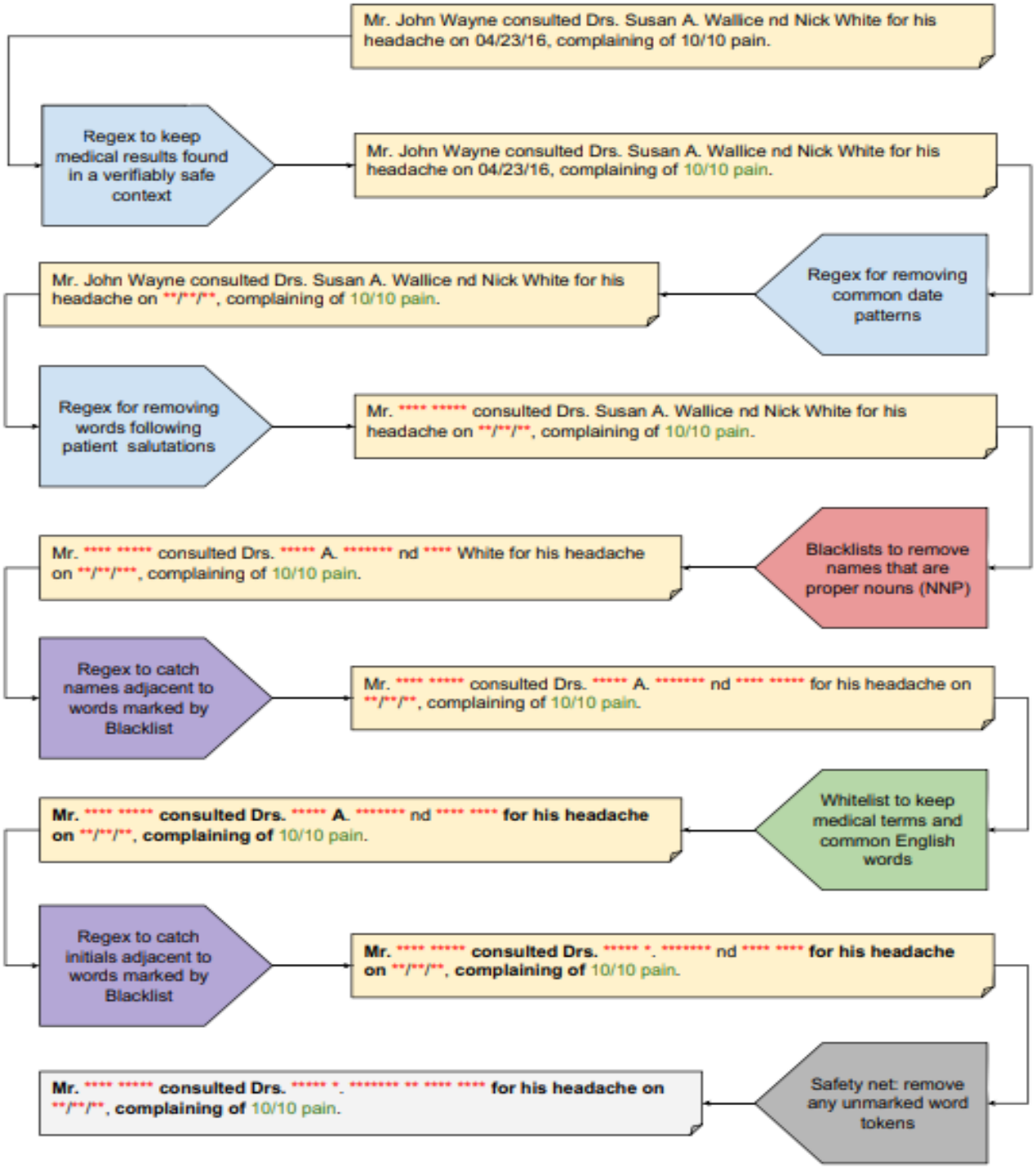


Figure 6-1: The Process of DE-Identification using Philter [149]

6.5.2. DE-identification System based on Artificial Neural Networks(ANNs)

Dernoncourt et al. [200] proposed the first de-identification system based on ANNs, which requires no hand-crafted features or rules. The system is constituted of three layers: 1. character-enhanced token embedding layer, 2. label prediction layer, and 3. label sequence optimization layer. The system compared with the state-of-the-art CRF models on two datasets: the i2b2 2014 corpus with a recall of 97.38 and a precision of 98.32, and an F1-score of 99.23, and the MIMIC corpus with a recall of 99.25 and a precision of 99.21

It demonstrated that combining RNNs with CRFs and training the model from end-to-end can achieve promising recall performance. The general shortcoming of such approaches in this context is that they depend heavily on labelled training data.

6.5.3. DE-identification system based on ANNs and CRF

Liu et al. model, detailed in [179], extended Dernoncourt et al [200] model by adding additional context features to the neural network. Liu et al. proposed a hybrid method that used an ensemble classifier to combine different methods: a CRF-based method, a bidirectional LSTM with hand-crafted features, a regular bidirectional LSTM, and a rule-based method. The system achieved a state-of-the-art F1 score of 98.28%.

6.6. Proposed Method

The proposed system consists of two key components that are integrated to de-identify unstructured textual data: 1) the core of Philter package [149] for PHI detection and 2) E-ART encryption algorithm [84] for replacement strategy. Figure 6-1 presents an illustration of the method. The steps and dataset details are provided in the following sections.

6.6.1. PHI Detection

To tackle the task of locating PHI, an overlapping pipeline of multiple state-of-the-art methods provided by Philter is used [149]. The pipeline includes pattern matching,

statistical modelling, blacklist, and whitelist, to detect PHI from free-text clinical notes. The detection process involves scanning the unstructured text line by line and dividing them into individual words. First, common words with a high probability of not being PHI are detected using pattern matching with a custom library of 133 "safe" regular expressions. Second, a customized library of 171 regular expressions is used to locate known PHI entities such as salutations, ID number, phone numbers, dates of birth, email address, and zip codes. In both scenarios, the regular expressions look for exact terms, phrases, and/or numbers to recognize matches using the immediate context surrounding each word. The algorithm used statistical modelling to determine the structure of each sentence and document in order to address the challenge of dealing with words that could be either safe or PHI such as White might be a name or an adjective. To exclude name that are proper noun, customized blacklist is used. And whitelist is used to preserve all the medical terms and common English word. At the end of the pipeline, a token has one of three potential labels: PHI, Non-PHI or unmarked.

6.6.2. Replacement strategy

To achieve the goal of developing reversible de-identification, all tokens marked as PHI, and unmarked tokens will be replaced with generated strings that can be retrieved. For this purpose, E-ART [84] the lightweight encryption algorithm proposed in chapter 4 has been used. The system will look for PHI entities labelled in the PHI detection phase and use the E-ART algorithm to encrypt them. E-ART uses the entity’s characters' index as a seed to generate a pseudo-random value as explain in chapter 4. This value will be used as a key to generate an encrypted string to replace the PHI entity. The encrypted PHI entity's start and end will be marked with asterisks to be recognised if recovery of the original data is needed, as shown in the de-identified text in Figure 6-2.

During the regeneration process of the original text, the de-identified text will be first scanned line by line, and all words that start and end with asterisks will be labelled as

encrypted PHI. Then all encrypted PHI will be decrypted by reversing the process of E-ART encryption as described in section 4.3.

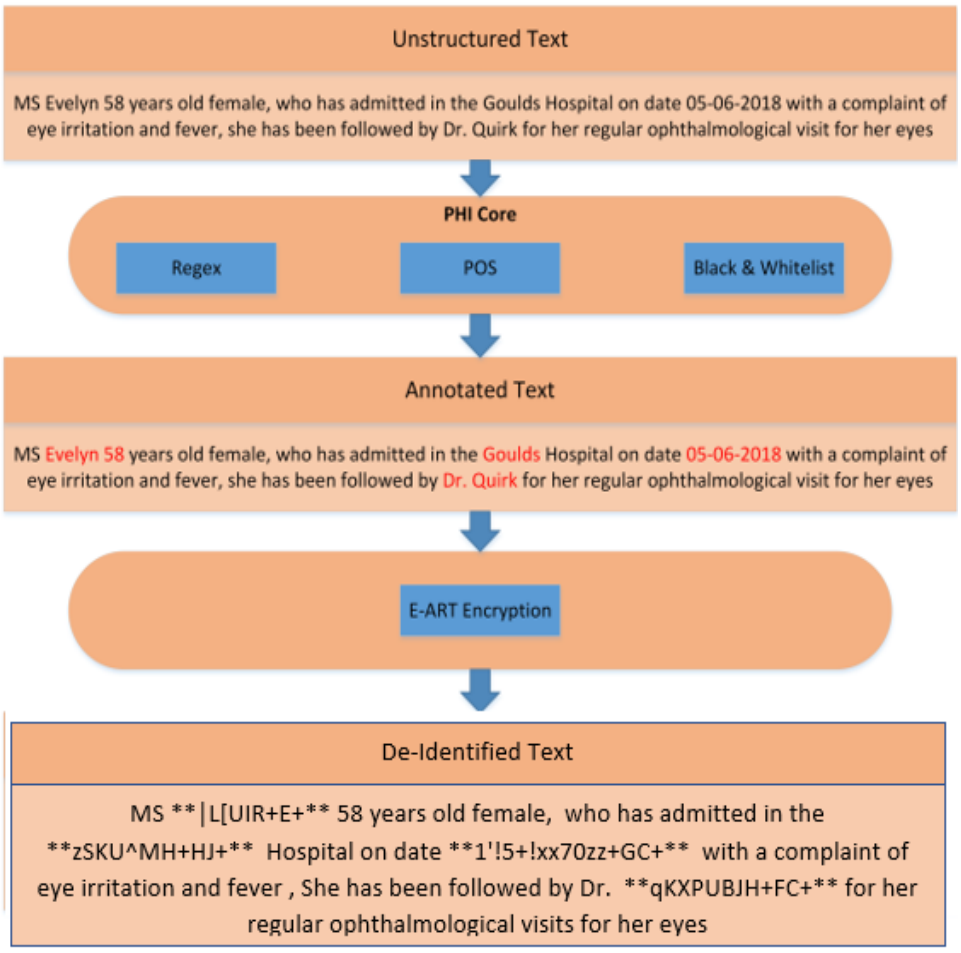


Figure 6-2: Integration of E-ART with Philter

6.7. Experiment Setup

This Section describes the experiments conducted to validate the proposed system. In particular, the Implementation setup is presented in Section 6.7.1, the dataset in Section 6.7.2 and the evaluation metrics are outlined in Section 6.7.3.

6.7.1. Implementation

The ARTPHIL package is written in Python 3.7 (32 bit) using Anaconda platform. The implementation used the core of Philter package and integrated it with E-ART

algorithm. The correctness of de-identification and regeneration has been tested on free text clinical note as shown in Table 6-3. The first column shows a made-up note for a doctor to one of his colleagues, second column shows de-identified of the note using ARTPHIL, and the third column shows the re-generated note.

Table 6-3: De-identification and Re-generated of Example Clinical Note

De-identification and Re-generation of Sample Clinical note		
Original note	**De-identified note**	**Re-generated note**
MS Evelyn 58 years old female, who has admitted in the Goulds Hospital on date 05-06-2018 with a complaint of eye irritation and fever, She has been followed by Dr. Quirk for her regular ophthalmological visits for her eyes. You can reach her via email (her address is Evelyn@gmail.com) or via phone: 998 785 6756. Sincerely, Elijah Hunt, MD	MS **|L[UIR+E+** 58 years old female , who has admitted in the **zSKU^MH+HJ+** Hospital on date **1'!5+!xx70zz+GC+** with a complaint of eye irritation and fever , She has been followed by Dr. **qKXPUBJH+FC+** for her regular ophthalmological visits for her eyes . You can reach her via email (her address is **GhJkJ]7YTaWU+5]RU**) or via phone **+yy+ [$!+*+zz+'U!+0%+T'M+ R**. Sincerely , **|VWWaXDIC** **zKSNDJI** , **s} **	MS Evelyn 58 years old female, who has admitted in the Goulds Hospital on date 05-06-2018 with a complaint of eye irritation and fever , She has been followed by Dr. Quirk for her regular ophthalmological visits for her eyes. You can reach her via email (her address is Evelyn @ gmail.com) or via phone 998 785 6756 . Sincerely, Elijah Hunt , MD

6.7.2. Dataset

The research used the i2b2/UTHealth 2014 de-identification corpus that was released by [157] as a part of the i2b2 National Center for Biomedical Computing for the NLP Shared Tasks Challenges, whose de-identification guidelines reported by [155] conform to the Safe Harbor criteria. All the PHI in the data set were hand labeled and obfuscated and replaced with realistic surrogated before the release. The data sets comprise 1304 longitudinal medical records of 296 patients with 2–5 records selected per patient and are officially divided into training and testing set. The training set contains 790 documents (including 269 for validation), while the testing set contains 514 documents. Each

document is a medical record in xml format, and the named entities within the documents are annotated as text spans with corresponding entity types. Table 6-4 presents a list of PHI distributions in the i2b2/UTHealth 2014 de-identification corpus. The data set is publicly available after the respective data use agreement is signed.

Table 6-4: PHI Distributions in the i2b2/UTHealth 2014 DE-Identification Corpus

PHI category: Subcategory	Training Data	Test Data	Total No. in Corpus
Name: Patient	1316	879	2195
Name: Doctor	2885	1912	4797
Name: Username	264	92	356
Profession	234	179	413
Location: hospital	1437	875	2312
Location: organisation	124	82	206
Location: Street	216	136	352
Location: City	394	260	654
Location: State	314	190	504
Location: Country	66	117	183
Location: Zip code	212	140	352
Location: Others	4	13	17
Age	1233	764	1997
Date	7507	4980	12487
Contact: Phone	309	215	524
Contact: Fax	8	2	10
Contact: email	4	1	5
Contact: URL	2	0	2
Contact: IP address: SSN	0	0	0
Id: Medical record	611	422	1033
Id: Health plan	1	0	1
Id: Account	0	0	0
Id:: Licence	0	0	0

$$, \beta = 2 \quad \text{(6-5)}$$

6.8. Results

Since the proposed de-identification system in this research and most existing de-identification system treat the PHI identification as a named entity recognition problem, the evaluation of the identification should be with the same metrics used in the named entity recognition and information retrieval literature [203]. In particular, the research used i2b2 annotated testing data set with all PHI categories discussed in detail in Section 6.7.2 and report performance primarily using three metrics: precision, recall, and f-measure. The evaluation started by removing all the annotation from the testing data and then de-identified it using the ARTPHIL model. after that, an evaluation script [8] that automatically compares the de-identified data with the annotated testing notes at the character level to quantify the PHI detection performance according to the evaluation matrices

Recall answers the question; "Did we find all that we were looking for?" and precision answers the question; "Did we only label what we were looking for?". A high recall is generally preferred over high precision, as it measures the percentage of PHI that is correctly identified, and the privacy of the data subjects is prioritized over a potential loss of document interpretability. To emphasize the sensitivity, F2 measure is calculated in addition to the F1 measure, which weighs recall (twice) higher than precision, as defined in eq. 6.4:

The overall recall and precision are computed as shown in Table 6-5, recall per tag as shown in Table 6-6 and Per-PHI-category recall as shown in Table 6-7 across the publicly available 514 notes 2014 i2b2 test corpus. In the six main categories, name, date, age, contact, and IDs received a high recall of above 0.9, but location others achieved recall below 90, as shown in Table 6-7. This category does not belong to the HIPAA PHI

[8] bayan6060/philter_eart (github.com)

categories. Consequently, the i2b2 PHI categories' overall result is worse than that of the HIPAA PHI categories [204]. However, the performance in several PHI types was good. For example, the recall of medical record, phone, email, fax and zip code achieved 100%.

From Table 6-5, it can be seen that the de-identification method achieves generally good results, with recall of 96.93%, a precision of 79.76%, F_1-score of 87.45% and F2-score of 92.92%. The recall of the method shows how likely it is for a PHI to be missed, and thus how likely it is for re-identification to occur, while the precision measures the number of false positives, thereby estimating the amount of information loss that is a result of applying the method. These results are in line with recent research as shown in Table 6-10 (see section 6.9.1). More importantly, patient names achieved 99.86% as only two names are missed out of 1447 names and for contact, medical record and zip code, which is the most directly identifying PHI, a good recall of 100% was achieved, while none of them was missed.

Table 6-5: Overall Model Performance

Metrics	Result
Precision	79.76%
Recall	96.93%
F1	87.45%
F2	92.92%

Table 6-6: Recall by Tag

Tags	Recall (%)	TPs	FNs
Medical record	100	721	0
Device	100	12	0
Username	98.91	91	1

Email	100	3	0
Fax	100	6	0
Zip code	100	143	0
Street	100	160	0
Location-Other	60	12	8
Patient	99.86	1445	2
Doctor	99.24	3272	25
City	98.54	338	2
Phone	100	407	0
State	96.10	197	8
Date	100	11880	0
Age	100	7	0
IDNUM	98.42	374	6

Table 6-7: Recall by PHI Category

Category	Recall (%)	TPs	FNs
Name	99.44	4718	27
Contact	100	416	0
ID	99.45	1113	6
Date	100	11880	0
Location	90.92	850	18
Age	100	7	0

6.8.1. Re-identification Risk

To estimate the re-identification risk of the proposed de-identification system, the conditional probability of a leak in identifiable information [203] is calculated as follow:

$$Pr\ (Reid, leak) = Pr(\text{re} - \text{identification}|leak) \times Pr(leak) \quad (6\text{-}6)$$

For a de-identification tool, a leak can occur if a PHI entity is not detected as it can be used to re-identify the individual whom data describe. So, it is assumed that:

$$P(Reid|leak) = 1 \quad (.6\text{-}7)$$

Recall measuring the percentage of PHIs that were correctly identified. Thus, the probability of a leak in a set of documents is directly related to recall, given by:

$$Pr(leak) = 1 - Recall \quad (6\text{-}8)$$

Table 6-8 lists the probability of risk re-identification for each PHI using this approach.

Table 6-8: The Probability of Risk Re-identification

Tags	Recall (%)	Probability of Re-Identification
Medical record	100	0
Device	100	0
Username	98.91	0.0109
Email	100	0
Fax	100	0
Zip	100	0
Street	100	0
Location-Other	60	0.40

Patient	99.86	0.0041
Doctor	99.24	0.0076
City	98.54	0.0146
Phone	100	0
State	96.10	0.039
Date	100	0
IDNUM	98.42	0.0158
Age	100	0
Average		0.0307

6.8.2. Execution Time

The run time of the integrated model is calculated using batches of 514 notes (3.2 MB) on a 4 core windows machine with 16 GB of RAM using the Python Time function, 'time', to estimate the feasibility of running ARTPHIL at a large scale data is conducted. Two experiments are conducted as follow; first, a single batch with a total size of 514 notes with a total size of 3.2MB, was run as single process and timed. Second, 20 batches of the 514 notes were run as single processes and timed. The amount of time necessary to run 514 notes as a single process was 2.5147seconds. The amount of real time necessary to run 20 batches of 514 notes, 10,280 notes total, was 54.675 seconds as shown in Table 6-9.

Table 6-9: The Execution time for ARTPHIL

Experiments	Time/ seconds
1- Single batch with 514 notes	2.5147
2- 20 batch with 514 notes	54.675

6.9. Effectiveness of ARTPHIL

This section demonstrates the effectiveness of ARTPHIL de-identification system. First, by comparing ARTPHIL results with the well-known and recent de-identification systems. Second by explaining how ARTHIL generates de-identified data that comply with the GDPR requirement for strong pseudonymization.

6.9.1. ARTPHIL and De-identification Systems:

The de-identification field has been dominated by two different approaches to designing de-identification models [149]. The first approach detects PHI using a rule-based system, while the second approach assigns probabilities of PHI words using statistics. Rule-based approaches mostly use regular expressions, whitelists, and blacklists to label PHI words, while statistical approaches use machine learning to learn patterns based on word context, traditionally using CRF and – more recently – using RNN. Rule-based approaches have higher recall, while statistical approaches have higher precision.

To evaluate the effectiveness of the ARTPHIL de-identification system, the researcher compared the model's performance with the following models: first, the two strongest rule-based de-identification competitors, Physionet [40] and Scrubber [202]; second, the Philter model [149] which combined rule-based and statistical approaches; and third, the state-of-the-art machine learning-based de-identification systems of Dernoncourt et al. [200] and Liu et al. [201], both of which are built on RNN architectures as shown in **Error! Reference source not found.**.

All six de-identification systems were trained on the i2b2 2014 training dataset and evaluated on the i2b2 2014 test dataset. In terms of recall, ARTPHIL outperformed Physionet with 69.84 % and Scrubber with 87.80 %, and achieved comparable results with other techniques. In terms of precision, ARTPHIL achieved a comparable result to Physionet of 89.49 %, Scrubber of 76.26%, and Philter of 78.58%. However, it achieved

a lower result compared to the machine learning-based models of Dernoncourt et al. model with 97.83% and Liu et al. model with 97.28%.

Table 6-10: Comparison of different de-identification models using I2b2 corpus

Model	Precision	Recall	F-measure
Rule-based Model			
Physionet [40]	89.49	69.84	73.05
Scrubber [202]	76.26	87.80	85.22
Hybrid Model			
Philter model [149]	78.58	99.92	94.77
ARTPHIL [205]	79.76	96.93	87.45
Machine Learning Model			
Dernoncourt et al.'s model [200]	97.83	97.92	97.88
Liu et al.'s model [201]	97.28	99.30	98.28

In summary, compared with existing rule-based models, ARTPHIL achieved higher recall and comparable precision. Compared with machine and deep learning models, ARTPHIL achieved comparable recall and lower precision. However, in de-identification systems, high recall is generally preferred over high precision because it measures the percentage of PHI that has been correctly identified, and the privacy of the data subject is prioritized over the potential loss of document utility. In addition, with machine learning approaches, it is difficult to determine why the model is committing errors, and additional training is needed when these approaches are applied for new datasets [193]. Furthermore, unlike existing techniques, which do not recover the PHI word after de-identification,

ARTPHIL replaces the PHI word with a dynamic pseudonym that can be recovered. The pseudonym changes for the same attribute, thereby disassociating the attribute from the data subject and preventing singling out and global linkability. This is an important advantage of ARTPHIL over existing de-identification models.

6.9.2. ARTPHIL and GDPR requirements:

The ARTPHIL system generates de-identified data that comply with the GDPR requirement for strong pseudonymization as follows:

- Other techniques used static pseudonymous for the identifiers which make them vulnerable to the mosaic effect . ARTPHIL uses dynamic encryption to de-identify the personal data. This dynamism helps reduce risk of re-identification via linkage attacks as it is difficult correlating data across available datasets

- ARTPHIL de-identifies all direct and indirect identifiers with a recall of 96.93%. This decreases the possibility of retaining any personal data that could re-identify the data subject, thus decreasing the potential violation of data subject privacy.

Data de-identification systems that meet the GDPR standard for data pseudonymization help to comply with GDPR as follows:

- They help of satisfy GDPR Article 5 [1] requirements as explained in chapter 2 for processing personal data so they comply with data minimisation, purpose and storage limitation principles.

- Using ARTPHIL to de-identify data helps to satisfy GDPR Article 6 [78]requirements for the lawful processing of personal data as an alternative for data subject consent.

- The processing of data de-identified using ARTPHIL helps organisations benefit from provisions under GDPR Articles 11(2) and 12(2) [70]. Controllers that do not control or have access to the key for re-identification can exempt subject rights for specific data, as mention in chapter 2.

6.10. Summary

The research objective of this thesis is to develop a reversible de-identification system to secure and prevent the unauthorized re-identification of unstructured health data. De-identification approaches include different forms of pseudonymization, which typically involves the removal or replacement of direct identifiers from the data set. However, data that could indirectly identify a person (quasi-identifiers) may be left in place. HIPAA privacy rules addressed this issue by specifying 18 data elements (include direct and indirect identifiers) that must be removed from data sets to be considered de-identified.

This chapter focused on addressing the following research question:

- How can the proposed encryption algorithm be used to de-identify unstructured textual data in order to improve individual privacy?

In this chapter, the stat-of-the-art of NER to extract personal identifying information from the unstructured text are used. To answer that question, Philter [149], the state-of-the-art tool for extracting personal identifiers from free-text is integrated with the encryption algorithm E-ART proposed in the previous chapter to detect and de-identify PHI specified under HIPAA guidelines. The i2b2/UTHealth 2014 de-identification corpus is used to evaluate the hybrid model. First, the correctness of de-identify and re-generate data is tested. Then, the performance is assessed using the NER evaluation metrics and the results are in line with recent research with recall of 96.93%, a precision of 79.76%, F_1-score of 87.45% and F2-score of 92.92%. More importantly, patient names achieved

%99.86 as only two names are missed out of 1447 names and for contact, medical record and zip code, which is the most directly identifying PHI, a good recall of 100% was achieved, while none of them was missed. Further, the probability of re-identification risk is estimated for each PHI category. Finally, the execution time for a large scale of data is calculated. The amount of time needed to de-identify 10,280 notes with (64 MB) note under is 54.675 seconds which indicate the suitability for ARTPHIL to be used with large data or delay-sensitive applications.

Chapter 7 : Conclusion and Future Works

This chapter summarises the thesis contributions, discusses the significant benefits of the proposed integrated de-identification system ARTPHIL, and discusses possible future research directions based on the current work.

7.1. Thesis Summary

The research was initiated because of privacy and security concerns triggered by the emergence of big data applications and technology and the fact that an increasing number of organisations rely on sharing, storing and analysing big data to fulfil their daily operations. The research investigates current privacy-preserving approaches and the new obligations imposed by GDPR to process personal data. The research aims to bridge the gap between (privacy regulations) GDPR obligations and the legitimate use of data through data de-identification to reduce the risk of disclosing data on subjects' identities.

The review shows that existing privacy-preserving approaches were developed long before GDPR requirements were established. For example, privacy-preserving data publishing (e.g., k-anonymity, l-diversity, and differential privacy) was developed primarily for structured data to release or share information with untrusted parties (e.g., aggregated data, statistical results or synthetic data). These approaches protect individual privacy by irreversibly removing identifiable information. However, they are not suitable for data processing within organisations to satisfy GDPR requirements and obligations. For example, for data protection by default that must be applied at the earliest opportunity (e.g., by pseudonymising data) to limit data use to the minimum extent to support each product or service. In addition, encrypting or masking the direct identifiers may fall short because of the risk of linkage re-identification. The gap in knowledge that resulted from this review was the need for a way to encrypt both direct and direct identifiers from unstructured text.

Further, standard encryption algorithms such as AES and DES may be robust pseudonymization techniques. However, these approaches rely on increasing the key size and the number of rounds to enhance security, which could negatively affect performance, especially for big data, or delay-sensitive applications. Section 2.6 discusses the new efforts that are being directed towards the design of new alternatives. The dynamic key theory introduced at the end of chapter 2, with an emphasis on the designing algorithm, relies on the idea of dynamic encryption rather than increasing the round and key size. The gap in knowledge that resulted from this review was the need to design a new cipher scheme that can overcome the disadvantages of existing ciphers, such as large keys and complex computations, to meet the requirement of processing large amounts of data rapidly. And support the design reversible de-identification model that satisfy GDPR requirement for strong

In this thesis, this issue is addressed by developing a reversible, fast model that use novel encryption algorithm to de-identify direct and indirect identifiers from unstructured textual data cost-effectively without compromising security.

The contributions of this research are summarized as follows:

1- a simple and novel encryption algorithm E-ART was proposed in chapter 4. The algorithm used the concept of the reflection balanced binary tree data structure and the ASCII table as a substitution method. The secret key is used to generate: 1) variable offset that is computed based on the E-ART tree's properties. 2) a dynamic key based on pseudo random number generators, character position and the secret key. The dynamic key varies for each character to reach the required confusion and diffusion features to prevent statistical attacks.

2- Chapter 5 evaluates the proposed algorithm against benchmark algorithms AES and DES and against defined criteria. The main objectives have been achieved by reducing

the execution time and the resources needed for the encryption process compared with the benchmark's algorithm and also to maintain its security level, to get high performance with high security. The chapter combined a series of experiments to evaluate security and performance. The results showed the following:

- The average encryption speed of E-ART is four times faster than AES and five times faster DES.
- The average memory consumption is comparable for all algorithms. However, the results indicate that E-ART consumes higher memory for large files.
- File size increased by only 5% compared to the plain-text file under E-ART, while it increased by around 31% for files encrypted using AES and DES.
- E-ART satisfies the avalanche effect criterion. The average avalanche effect of E-ART was 50.11% compared to 49.2% for AES and 49.3% for DES when a single bit changed in the key.
- E-ART satisfies the BIC criteria. The results indicate very low dependence between the two avalanche variables. E-ART yielded a correlation value of 0.1863 compared to 0.1818 for AES and 0.1847 for DES.
- The histogram analysis performed on the plaintext and ciphertext showed a higher level of uniform distribution for AES and DES than E-ART. However, the histogram of the E-ART is distributed in a relatively uniform way, and it differs significantly from that of the plain-text histogram.
- The analysis of the dynamic key produced by pseudo-random number generators using the character position and secret key indicates that the sequences are truly random.

3- The algorithm proposed in chapter 4 and validated in chapter 5 has been used as a replacement strategy to develop a fast and reversible de-identification system to de-identify all PHI specified by HIPAA from unstructured text in chapter 6. The de-identification system used the overlapping pipeline provided by Philter [149]. It includes pattern matching, statistical modelling, blacklisting and whitelisting to detect all personal data (including direct and indirect identifiers) from free-text and encrypted using the E-ART algorithm.

4- The system was evaluated using the i2b2 2014 testing corpus annotated to comply with HIPAA guidelines. ARTPHIL achieved an overall F2 score of 92.92%, with recall of 96.93% and the precision of 79% recall of 96.93%, to detect and encrypt all personal information specified under HIPAA guidelines. As the main goal of the research is to reduce the privacy risk of data subjects, high recall is prioritised over high precision. The amount of time needed to de-identify 10,280 patient records was 54.675 seconds, which indicates the suitability of using ARTPHIL with large-scale data. Comparing with existing rule-based models, ARTPHIL achieved higher recall and comparable precision. Compared with machine and deep learning models, ARTPHIL achieved comparable recall and lower precision

5- Pseudonymization that satisfy GDPR requirements helps to relax certain data controller obligations as mentioned in section ( ) . The ARTPHIL system generates de-identified data that comply with the GDPR requirement for strong pseudonymization as follows:

- ARTPHIL uses dynamic encryption to de-identify the personal data. This dynamism helps reduce risk of re-identification via linkage attacks as it is difficult correlating data across available datasets. Unlike other techniques used static pseudonymous for the identifiers which make them vulnerable to the mosaic effect .

- ARTPHIL de-identifies all direct and indirect identifiers with a recall of 96.93%. This decreases the possibility of retaining any personal data that could re-identify the data subject, thus decreasing the potential violation of data subject privacy.

7.2. Suggestions and future direction

Several important issues regarding the design of privacy-preserving techniques have been addressed by this research. The research provides both theoretical and empirical research on the domain. The main contribution is the proposed new de-identification system that de-identifies data subjects, thus reducing risk to the privacy of data subject while making beneficial use of data. In the process, several challenges were encountered and addressed. This section highlights some of the remaining challenges, which suggest directions for future research.

This research can be extended in different directions, including:

E-ART:

Further development of the E-ART algorithm can be performed as follow:

- Extending the E-ART algorithm to encrypt other types of data.

The proposed E-ART algorithm encrypts unstructured textual data using a reflection balanced binary tree along with an ASCII table. Extending the algorithm so that it can encrypt other types of data, including images, sounds and videos, will be of great importance. This could be done by adding a binary conversion step to the encryption processes.

- Code optimization

The performance of E-ART (in terms of speed and memory consumption) could be further improved by optimizing the algorithm code. For example, by trying different data programming structures.

ARTPHIL:

- Improving recall

The PHI removal can be further optimised using a crowd-sourcing approach with lots of exposure to many hospitals and notes.

- Extending the implementation to other domains

The ARTPHIL de-identification system is highly customisable, so it can be used not only in the healthcare domain but also in other domains, such as finance and education. This can be achieved by customising its implementation (e.g. adding a custom library), depending on the quasi-identifiers to be encrypted on the specific domain.

- Reducing information loss by applying partial de-identification

Partial de-identification to reduce information loss could be an interesting direction in future research. By partial de-identification, it means that only a part of the PHI is encrypted, leaving the remaining corresponding tokens to be changed from PHI to safe. For example, consider part of the date. (e.g. 30/9/2021 → X/X/2021) or most of a name (e.g. David A Smith → XXX A XXXX).

- Introducing different levels of de-identification

Instead of de-identifying data based on HIPAA guidelines, the data could be identified based on the user processing the data. For example, the same data could represent one level of de-identification to one user and another level to another user.

7.3. Summary

This chapter has summarised the research done in this thesis, explained the main contributions, identified significant and exciting new directions for further investigation. More specifically, this research has reviewed relevant literature to find a knowledge gap in privacy-preservation approaches and proposed a novel and lightweight encryption algorithm. Then, the proposed algorithm was used to develop a new de-identification system for unstructured textual data for minimizing the risk of re-identification and allowing legitimate use of data

The author hopes that future research comes from this thesis, and this chapter makes some suggestions in that regard. Indeed, E- ART encryption algorithm[9] and ARTPHIL de-identification system[10] are obvious candidates for further study and so that they made available in the GitHub repository.

[9] bayan6060/eart1 (github.com)
[10] bayan6060/philter_eart (github.com)

References

[1] "Art. 5 GDPR – Principles relating to processing of personal data - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-5-gdpr/. [Accessed: 27-Sep-2021].

[2] "Art. 89 GDPR – Safeguards and derogations relating to processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-89-gdpr/. [Accessed: 27-Sep-2021].

[3] Office of the Secretary, "Standards for Privacy of Individually Identifiable Health Information; Final Rule," *Fed. Regist.*, vol. 67, no. 157, pp. 1–93, 2002.

[4] A. Hundepool *et al.*, *Statistical disclosure control*, vol. 2. Wiley New York, 2012.

[5] B. C. M. Fung, K. E. Wang, R. U. I. Chen, and P. S. Yu, "Privacy-Preserving Data Publishing : A Survey of Recent Developments," *ACM Comput. Surv.*, vol. 42, no. 4, pp. 1–53, 2010.

[6] T. S. Chen, W. Bin Lee, J. Chen, Y. H. Kao, and P. W. Hou, "Reversible privacy preserving data mining: A combination of difference expansion and privacy preserving," *J. Supercomput.*, vol. 66, no. 2, pp. 907–917, 2013.

[7] T. P. Hong, L. H. Tseng, and B. C. Chien, "Mining from incomplete quantitative data by fuzzy rough sets," *Expert Syst. Appl.*, vol. 37, no. 3, pp. 2644–2653, 2010.

[8] J. F. J. M. Silva, E. Pinho, E. Monteiro, J. F. J. M. Silva, and C. Costa, "Controlled searching in reversibly de-identified medical imaging archives," *J. Biomed. Inform.*, vol. 77, no. November 2017, pp. 81–90, 2018.

[9] M. Verhenneman, G., Claes, K., Derèze, J.J., Herijgers, P., Mathieu, C., Rademakers, F.E., Reyda, R., & Vanautgaerden, "How GDPR Enhances Transparency and Fosters Pseudonymisation in Academic Medical Research," *Eur. J. Health Law*, vol. 27, no. 1, pp. 35–75, 2020.

[10] W. Landi and R. B. Rao, "Secure De-identification and Re-identification," in *AMIA Annual Symposium Proceedings*, 2003, vol. 65, no. 250, p. 905.

[11] M. Yamac, M. Ahishali, N. Passalis, J. Raitoharju, B. Sankur, and M. Gabbouj, "Reversible Privacy Preservation using Multi-level Encryption and Compressive Sensing."

[12] J. R. Gulcher and K. Kristj, "Protection of privacy by third-party encryption in genetic research in Iceland," pp. 739–742, 2000.

[13] I. Kuzminykh, A. Carlsson, M. Yevdokymenko, and V. Sokolov, "Investigation of the IoT Device Lifetime with Secure Data Transmission," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11660 LNCS, no. November, pp. 16–27, 2019.

[14] J. L. Hernández-Ramos *et al.*, "Protecting Personal Data in IoT Platform Scenarios

Through Encryption-based Selective Disclosure," *Comput. Commun.*, vol. 130, no. July, pp. 20–37, 2018.

[15] J. Zandy, "Universal declaration of human rights," *Radic. Teach.*, vol. 113, no. December, pp. 54–55, 2019.

[16] M. Langheinrich, "Privacy in Ubiquitous Computing," *Ubiquitous Comput. Fundam.*, pp. 95–159, 2009.

[17] T. Wright, "Security, privacy, and anonymity," *XRDS Crossroads, ACM Mag. Students*, vol. 11, no. 2, pp. 5–5, 2004.

[18] G. L. Alessandro. Acquisti, Laura Brandimarte, "Privacy and human behavior in the age of information," *New Electron.*, vol. 40, no. 16, pp. 49–50, 2015.

[19] S. Yu and S. Member, "Big Privacy : Challenges and Opportunities of Privacy Study in the Age of Big Data," *IEEE Access*, vol. 4, pp. 2751–2763, 2016.

[20] D. Banisar and S. Davies, "Global trends in privacy protection : An international survey of privacy, data protection, and surveillance laws and developments," *John Marshall J. Comput. Inf. Law*, vol. 18, no. 1, pp. 1–111, 1999.

[21] T. C. Clark and A. F. Westin, "Privacy and Freedom," *Calif. Law Rev.*, vol. 56, no. 3, p. 911, 1968.

[22] E. Bertino, D. Lin, and W. Jiang, "A Survey of Quantification of Privacy Preserving Data Mining Algorithms," pp. 183–205, 2008.

[23] W. Stallings, *Computer Security Principles and Practices*. Upper Saddle River, NJ, USA, 2014.

[24] R. Gavison, "Privacy and the Limits of Law," *Yale Law J.*, vol. 89, no. 3, p. 51, 1980.

[25] L. Sweeney, "Uniqueness of Simple Demographics in the U.S. Population, LIDAP-WP4," *Forthcom. B. entitled, Identifiability Data.*, pp. 1–34, 2000.

[26] S. L. Garfinkel, "De-Identification of Personal Information," *Nistir 8053*, pp. 1–46, 2015.

[27] F. Deldar and M. Abadi, "PDP-SAG: Personalized Privacy Protection in Moving Objects Databases by Combining Differential Privacy and Sensitive Attribute Generalization," *IEEE Access*, vol. 7, pp. 85887–85902, 2019.

[28] Y. N. Fakeeroodeen and Y. Beeharry, "Hybrid Data Privacy and Anonymization Algorithms for Smart Health Applications," *SN Comput. Sci.*, vol. 2, no. 2, pp. 1–10, 2021.

[29] D. Lambert, "Measures of disclosure risk and harm," *J. Off. Stat.*, vol. 9, no. 2, pp. 313–331, 1993.

[30] Article 29 Data Protection Working Party, "Opinion 05/2014 on Anonymisation Techniques," *Work. Party Opin.*, no. April, pp. 1–37, 2014.

[31] R. Hu, S. Stalla-Bourdillon, M. Yang, V. Schiavo, and V. Sassone, "Bridging Policy, Regulation and Practice? A techno-legal Analysis of Three Types of Data in the GDPR Runshan Hu, Sophie Stalla-Bourdillon, Mu Yang, Valeria Schiavo and Vladimiro Sassone," *Data Prot. Priv. Age Intell. Mach.*, p. 39, 2017.

[32] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, “Privacy-preserving data publishing,” *ACM Comput. Surv.*, vol. 42, no. 4, 2010.

[33] A. Mehmood, I. Natgunanathan, Y. Xiang, G. Hua, S. Guo, and S. Member, “Protection of big data privacy,” *IEEE Access*, vol. 4, pp. 1821–1834, 2016.

[34] P. Samarati and L. Sweeney, “Protecting Privacy when Disclosing Information : k - Anonymity and Its Enforcement through Generalization and Suppression 1 Introduction,” pp. 1–19.

[35] E. T. Kun Liu, “Towards Identity Anonymization on Graphs,” *Proc. 2008 ACM SIGMOD Int.*, vol. 36, no. 20, pp. 1627–1638, 2008.

[36] N. Li, T. Li, and S. Venkatasubramanian, “t-Closeness: Privacy Beyond k-Anonymity and L-Diversity,” *Data Eng. 2007. ICDE 2007. IEEE 23rd Int. Conf.*, pp. 106–115, 2007.

[37] Q. Wang, Z. Xu, and S. Qu, “An enhanced k-anonymity model against homogeneity attack,” *J. Softw.*, vol. 6, no. 10, pp. 1945–1952, 2011.

[38] T. Li, N. Li, and J. Zhang, “Modeling and integrating background knowledge in data anonymization,” *Proc. - Int. Conf. Data Eng.*, pp. 6–17, 2009.

[39] L. S. Pierangela Samarati, “Generalizing Data to Provide Anonymity when Disclosing Information,” *PODS*, vol. 23, no. 3, pp. 27–28, 1998.

[40] I. Neamatullah *et al.*, “Automated De-identification of Free-text Medical Records,” *BMC Med. Inform. Decis. Mak.*, vol. 8, pp. 1–17, 2008.

[41] X. B. Li and J. Qin, “Anonymizing and sharing medical text records,” *Inf. Syst. Res.*, vol. 28, no. 2, pp. 332–352, 2017.

[42] N. Memon, “Anonymizing Large Transaction Data Using MapReduce,” no. January. School of Computer Science & Informatics Cardiff University, Cardiff, 2016.

[43] B. Alabdullah, “Rise of Big Data; Issues and Challenges,” *2018 21st Saudi Comput. Soc. Natl. Comput. Conf.*, pp. 1–6, 2018.

[44] C. A. Kushida, D. A. Nichols, R. Jadrnicek, R. Miller, J. K. Walsh, and K. Griffin, “Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies,” *Med Care*, vol. 50, no. July, 2012.

[45] UiO, “Personal Data, Anonymisation & Pseudonymisation under European Data Protection Law A Comparison of the DPD and the GDPR on the Example of Cloud Computing,” *Univ. Oslo Fac. Law*, p. 63, 2016.

[46] “Art. 4 GDPR – Definitions - General Data Protection Regulation (GDPR).” [Online]. Available: https://gdpr-info.eu/art-4-gdpr/. [Accessed: 27-Sep-2021].

[47] “Recital 26 - Not Applicable to Anonymous Data - General Data Protection Regulation (GDPR).” [Online]. Available: https://gdpr-info.eu/recitals/no-26/. [Accessed: 27-Sep-2021].

[48] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3876 LNCS, pp. 265–284, 2006.

[49] P. Jain, M. Gyanchandani, and N. Khare, “Big Data Privacy : A Technological Perspective and Review,” *J. Big Data*, vol. 3, no. 1, 2016.

[50] H. Vaghashia, A. Ganatra, and P. U. Patel, “A Survey: Privacy Preservation Techniques in Data Mining,” *Int. J. Comput. Appl.*, vol. 119, no. 4, pp. 975–8887, 2015.

[51] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, “SCiFI - A system for secure face identification,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 239–254, 2010.

[52] M. L. D. Ronal L. Rivest, Len Adleman, “ON DATA BANKS AND PRIVACY HOMOMORPHISMS,” *Found. Secur. Comput.*, vol. 4, no. 3, pp. 169–180, 1978.

[53] P. Venkateswarlu, B. Manasa, and K. Srikanth, “An Expensive Study of Homomorphic Encryption to Secure Cloud Data,” *Int. J. Comput. Sci. Eng.*, vol. 7, no. 4, pp. 765–770, 2019.

[54] J. Bringer, H. Chabanne, M. Favre, A. Patey, T. Schneider, and M. Zohner, “GSHADE: Faster privacy-preserving distance computation and biometric identification,” *IH MMSec 2014 - Proc. 2014 ACM Inf. Hiding Multimed. Secur. Work.*, pp. 187–198, 2014.

[55] M. Sepehri, S. Cimato, and E. Damiani, “Privacy-preserving query processing by multi-party computation,” *Comput. J.*, vol. 58, no. 10, pp. 2195–2212, 2015.

[56] M. M. Groat, W. He, and S. Forrest, “KIPDA: k-Indistinguishable Privacy-preserving Data Aggregation in Wireless Sensor Networks,” in *Proceedings IEEE INFOCOM*, 2011, pp. 2024–2032.

[57] Z. Yang, M. Yang, and B. Ning, “User relationship privacy protection on trajectory data,” in *Lecture Notes in Electrical Engineering*, 2020, vol. 517, pp. 1038–1045.

[58] J. V. Bibal Benifa and G. Venifa Mini, “Privacy Based Data Publishing Model for Cloud Computing Environment,” *Wirel. Pers. Commun.*, vol. 113, no. 4, pp. 2215–2241, 2020.

[59] B. S. Bhati, J. Ivanchev, I. Bojic, A. Datta, and D. Eckhoff, “Utility-Driven k-Anonymization of Public Transport User Data,” *IEEE Access*, vol. 9, pp. 23608–23623, 2021.

[60] B. Bamba, L. Liu, P. Pesti, and T. Wang, “Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid,” *Proceeding 17th Int. Conf. World Wide Web 2008, WWW’08*, pp. 237–246, 2008.

[61] A. M. Olawoyin, C. K. L. B, and R. Choudhury, “Privacy-Preserving Spatio-Temporal Patient Data Publishing,” 2020, pp. 407–416.

[62] Corporatefinanceinstitute.com, *Data Pseudonymisation Techniques: Advanced Techniques & Technical analysis of cybersecurity measures in data*, no. January. 2021.

[63] P. Jagwani, S. Tiwari, and S. Kaushik, “Using middleware as a certifying authority in LBS applications,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7108 LNCS, no. December, pp. 242–255, 2011.

[64] M. B. Hossain, I. Natgunanathan, Y. Xiang, and Y. Zhang, “Cost-Friendly Differential Privacy of Smart Meters Using Energy Storage and Harvesting Devices,” *IEEE Trans. Serv. Comput.*, no. May, 2021.

[65] R. Mendes and J. P. Vilela, “Privacy-Preserving Data Mining: Methods, Metrics, and

Applications," *IEEE Access*, vol. 5, pp. 10562–10582, 2017.

[66] T. Zhu, G. Li, W. Zhou, and P. S. Yu, "Differentially Private Spatial Crowdsourcing," *Adv. Inf. Secur.*, vol. 69, pp. 173–189, 2017.

[67] R. ji Huang, Q. Ye, and M. C. Li, "Clustering-Anonymization-Based Differential Location Privacy Preserving Protocol in WSN," *Commun. Comput. Inf. Sci.*, vol. 1120 CCIS, pp. 177–193, 2019.

[68] K. Aulakh and R. K. Ramachandran, "A Detailed Survey of Fully Homomorphic Encryption Standards to Preserve Privacy over Cloud Communications," *Indo - Taiwan 2nd Int. Conf. Comput. Anal. Networks, Indo-Taiwan ICAN 2020 - Proc.*, pp. 207–211, 2020.

[69] G. Bréart *et al.*, "Letters to the editor: Directive of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data," *Int. J. Epidemiol.*, vol. 24, no. 2, pp. 462–463, 1995.

[70] "Art. 11 GDPR – Processing which does not require identification - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-11-gdpr/. [Accessed: 27-Sep-2021].

[71] "Art. 15 GDPR – Right of access by the data subject - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-15-gdpr/. [Accessed: 27-Sep-2021].

[72] "Art. 16 GDPR – Right to rectification - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-16-gdpr/. [Accessed: 27-Sep-2021].

[73] "Art. 17 GDPR – Right to erasure ('right to be forgotten') - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-17-gdpr/. [Accessed: 27-Sep-2021].

[74] "Art. 18 GDPR – Right to restriction of processing - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-18-gdpr/. [Accessed: 27-Sep-2021].

[75] "Art. 20 GDPR – Right to data portability - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-20-gdpr/. [Accessed: 27-Sep-2021].

[76] "Art. 13 GDPR – Information to be provided where personal data are collected from the data subject - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-13-gdpr/. [Accessed: 27-Sep-2021].

[77] "Art. 14 GDPR – Information to be provided where personal data have not been obtained from the data subject - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-14-gdpr/. [Accessed: 27-Sep-2021].

[78] "Art. 6 GDPR – Lawfulness of processing - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-6-gdpr/. [Accessed: 27-Sep-2021].

[79] "Art. 25 GDPR – Data protection by design and by default - General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/art-25-gdpr/. [Accessed:

27-Sep-2021].

[80] “Art. 33 GDPR – Notification of a personal data breach to the supervisory authority - General Data Protection Regulation (GDPR).” [Online]. Available: https://gdpr-info.eu/art-33-gdpr/. [Accessed: 27-Sep-2021].

[81] “Art. 34 GDPR – Communication of a personal data breach to the data subject - General Data Protection Regulation (GDPR).” [Online]. Available: https://gdpr-info.eu/art-34-gdpr/. [Accessed: 27-Sep-2021].

[82] “Recital 28 - Introduction of Pseudonymisation - General Data Protection Regulation (GDPR).” [Online]. Available: https://gdpr-info.eu/recitals/no-28/. [Accessed: 27-Sep-2021].

[83] B. A. Forouzan, *Cryptography and Network Security*, vol. 1025, no. 1. McGraw-Hill, Inc.., 2007.

[84] B. Alabdullah, N. Beloff, and M. White, “E-ART: A New Encryption Algorithm Based on the Reflection of Binary Search Tree,” *Cryptography*, vol. 5, no. 1, p. 4, 2021.

[85] Y. Zheng and X. M. Zhang, “On Relationships Among Avalanche, Nonlinearity, and Correlation Immunity,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 1976, pp. 470–482, 2000.

[86] J. Lee, N. Sultana, F. Yi, and I. Moon, “Avalanche and Bit Independence Properties of Photon-Counting Double Random Phase Encoding in Gyrator Domain,” *Curr. Opt. Photonics*, vol. 2, no. 4, pp. 368–377, 2018.

[87] A. F. W. S.E.Tavares, “On the Design of S-Boxes,” *Proc. IRE*, vol. 42, no. 8, pp. 1262–1267, 1954.

[88] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. 2013.

[89] S. S. Omran, A. S. Al-Khalid, and D. M. Al-Saady, “A Cryptanalytic Attack on Vigenère Cipher using Genetic Algorithm,” *2011 IEEE Conf. Open Syst. ICOS 2011*, no. September, pp. 59–64, 2011.

[90] O. Calin, “Statistics and Machine Learning Experiments in English and Romanian Poetry,” *Sci*, vol. 2, no. 4, p. 92, 2020.

[91] Z. Hercigonja, “Comparative Analysis of Cryptographic Algorithms,” *Int. J. Digit. Technol. Econ.*, vol. 1, no. 2, pp. 127–134, 2016.

[92] D. Cryptosystems and E. Biham, “Differential Crypt Analysis of DES-Like Cryptosystems,” *J. Cryptol.*, vol. 4, no. 1, pp. 2–72, 1993.

[93] A. Bogdanov and C. Rechberger, “A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6544 LNCS, pp. 229–240, 2011.

[94] H. Noura, L. Sleem, M. Noura, M. M. Mansour, A. Chehab, and R. Couturier, “A new Efficient Lghtweight and Secure Image Cipher Scheme,” *Multimed. Tools Appl.*, vol. 77, no. 12, pp. 15457–15484, 2018.

[95] Biryukov Alex and C. De Cannière, “DATA ENCRYPTION STANDARD (DES),”

Encycl. Cryptogr. Secur., pp. 295–301, 2011.

[96] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999.

[97] M. Masoud, I. Jannoud, A. Ahmad, and H. Al-Shobaky, "The power consumption cost of data encryption in smartphones," *2015 Int. Conf. Open Source Softw. Comput. OSSCOM 2015*, no. August 2017, 2016.

[98] H. N. Noura, A. Chehab, and R. Couturier, "Efficient & Secure Cipher Scheme with Dynamic key-Dependent Mode of Operation," *Signal Process. Image Commun.*, vol. 78, no. January, pp. 448–464, 2019.

[99] G. Bansod, N. Raval, and N. Pisharoty, "Implementation of a new lightweight encryption design for embedded security," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 1, pp. 142–151, 2015.

[100] S. Aljawarneh, M. B. Yassein, and W. A. Talafha, "A Multithreaded Programming Approach for Multimedia Big Data: Encryption System," *Multimed. Tools Appl.*, vol. 77, no. 9, pp. 10997–11016, 2018.

[101] O. A. Dawood, A. M. Sagheer, and S. S. Al-Rawi, "Design Large Symmetric Algorithm for Securing Big Data," *Proc. - Int. Conf. Dev. eSystems Eng. DeSE*, vol. 2018-Septe, pp. 123–128, 2019.

[102] A. H. Al-Omari, "Lightweight Dynamic Crypto Algorithm for Next Internet Generation," *Eng. Technol. Appl. Sci. Res.*, vol. 9, no. 3, pp. 4203–4208, 2019.

[103] P. Derbez, P. A. Fouque, and J. Jean, "Improved Key Recovery Attacks on Reduced-Round AES in the Single-key Setting," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7881 LNCS, pp. 371–387, 2013.

[104] H. Mala, M. Dakhilalian, V. Rijmen, and M. Modarres-Hashemi, "Improved Impossible Differential Cryptanalysis of 7-round AES-128," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6498 LNCS, pp. 282–291, 2010.

[105] K. Gai, M. Qiu, H. Zhao, and J. Xiong, "Privacy-Aware Adaptive Data Encryption Strategy of Big Data in Cloud Computing," in *In 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2016, pp. 273–278.

[106] D. Puthal, X. Wu, N. Surya, R. Ranjan, and J. Chen, "SEEN: A selective encryption method to ensure confidentiality for big sensing data streams," *IEEE Trans. Big Data*, vol. 5, no. 3, pp. 379–392, 2019.

[107] O. A. Khashan and M. AlShaikh, "Edge-based lightweight selective encryption scheme for digital medical images," *Multimed. Tools Appl.*, no. September, 2020.

[108] D. Engel, T. Stütz, and A. Uhl, "Format-compliant JPEG2000 encryption with combined packet header and packet body protection," *MM Sec'07 - Proc. Multimed. Secur. Work. 2007*, pp. 87–96, 2007.

[109] M. C. Alipour, B. D. Gerardo, and R. P. Medina, "A secure image encryption architecture based on pseudorandom number generator and chaotic logistic map," *ACM Int. Conf. Proceeding Ser.*, pp. 154–159, 2019.

[110] H. H. Ngo, X. Wu, P. D. Le, C. Wilson, and B. Srinivasan, "Dynamic Key Cryptography

and Applications," *Int. J. Netw. Secur.*, vol. 10, no. 3, pp. 161–174, 2010.

[111] D. C. Hassan NOURA, "Implementation and Practical Problems of Chaos-based Cryptography Revisited," 2017.

[112] J. Sen Teh, M. Alawida, and Y. C. Sii, "Implementation and practical Problems of Chaos-based Cryptography Revisited," *J. Inf. Secur. Appl.*, vol. 50, no. August 2019, 2020.

[113] C. Chunka, R. S. Goswami, and S. Banerjee, "An Efficient Mechanism to Generate Dynamic keys Based on Genetic Algorithm," *Secur. Priv.*, p. e37, 2018.

[114] H. Noura, A. Chehab, and R. Couturier, "Lightweight Dynamic Key-Dependent and Flexible Cipher Scheme for IoT Devices," *IEEE Wirel. Commun. Netw. Conf. WCNC*, vol. 2019-April, pp. 1–8, 2019.

[115] Ü. Çavuşoğlu, S. Kaçar, A. Zengin, and I. Pehlivan, "A novel Hybrid Encryption Algorithm Based on Chaos and S-AES Algorithm," *Nonlinear Dyn.*, vol. 92, no. 4, pp. 1745–1759, 2018.

[116] X. Chai, K. Yang, and Z. Gan, "A new chaos-based image encryption algorithm with dynamic key selection mechanisms," *Multimed. Tools Appl.*, vol. 76, no. 7, pp. 9907–9927, 2017.

[117] O. Jallouli, S. El Assad, M. Chetto, and R. Lozi, "Design and Analysis of Two Stream Ciphers Based on Chaotic Coupling and Multiplexing Techniques," *Multimed. Tools Appl.*, vol. 77, no. 11, pp. 13391–13417, 2018.

[118] W. Wen, Y. Zhang, M. Su, R. Zhang, J. xin Chen, and M. Li, "Differential Attack on A hyper-Chaos-based Image Cryptosystem with A classic bi-Modular Architecture," *Nonlinear Dyn.*, vol. 87, no. 1, pp. 383–390, 2017.

[119] J. Sen Teh, M. Alawida, and Y. C. Sii, "Implementation and Practical Problems of Chaos-based Cryptography Revisited," *J. Inf. Secur. Appl.*, vol. 50, no. November, 2020.

[120] L. Ding, C. Liu, Y. Zhang, and Q. Ding, "A new Lightweight Stream Cipher Based on Chaos," *Symmetry (Basel).*, vol. 11, no. 7, pp. 1–12, 2019.

[121] A. Arab, M. J. Rostami, and B. Ghavami, "An image encryption method based on chaos system and AES algorithm," *J. Supercomput.*, vol. 75, no. 10, pp. 6663–6682, 2019.

[122] X. Chai, X. Fu, Z. Gan, Y. Lu, and Y. Chen, "A color Image Cryptosystem Based on Dynamic DNA Encryption and Chaos," *Signal Processing*, vol. 155, pp. 44–62, 2019.

[123] L. Shi, Y. Wang, R. Jia, T. Peng, J. Jiang, and S. Zhu, "Research of Lightweight Encryption Algorithm Based on AES and Chaotic Sequences for Narrow-Band Internet of Things," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2019, vol. 294 LNCIST, pp. 267–280.

[124] S. O. Tuam, "Text Encryption Approach using DNA Computation and Chaotic Indexing," *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 8, pp. 4654–4658, 2020.

[125] M. Babaei, "A novel text and image encryption method based on chaos theory and DNA computing," *Nat. Comput.*, vol. 12, no. 1, pp. 101–107, 2013.

[126] Q. Zhang, L. Guo, and X. Wei, "A novel image fusion encryption algorithm based on

DNA sequence operation and hyper-chaotic system," *Optik (Stuttg).*, vol. 124, no. 18, pp. 3596–3600, 2013.

[127] J. Lü, G. Chen, D. Cheng, and S. Celikovsky, "Bridge the gap between the Lorenz system and the Chen system," *Int. J. Bifurcat. Chaos*, vol. 12, no. 12, pp. 2917–2926, 2002.

[128] X. Wang and Y. Su, "An Audio Encryption Algorithm Based on DNA Coding and Chaotic System," *IEEE Access*, vol. 8, pp. 9260–9270, 2020.

[129] L. Liu, Q. Zhang, and X. Wei, "A RGB image encryption algorithm based on DNA encoding and chaos map," *Comput. Electr. Eng.*, vol. 38, no. 5, pp. 1240–1248, 2012.

[130] F. Özkaynak, A. B. Özer, and S. Yavuz, "Security Analysis of An Image Encryption Algorithm Based on Chaos and DNA Encoding," *2013 21st Signal Process. Commun. Appl. Conf. SIU 2013*, no. l, pp. 3–6, 2013.

[131] Y. Liu, J. Tang, and T. Xie, "Cryptanalyzing a RGB image encryption algorithm based on DNA encoding and chaos map," *Opt. Laser Technol.*, vol. 60, pp. 111–115, 2014.

[132] X. Huang and G. Ye, "An image encryption algorithm based on hyper-chaos and DNA sequence," *Multimed. Tools Appl.*, vol. 72, no. 1, pp. 57–70, 2014.

[133] Q. Zhang, L. Guo, and X. Wei, "Image encryption using DNA addition combining with chaotic maps," *Math. Comput. Model.*, vol. 52, no. 11–12, pp. 2028–2035, 2010.

[134] V. R. Folifack Signing, T. Fozin Fonzin, M. Kountchou, J. Kengne, and Z. T. Njitacke, *Chaotic Jerk System with Hump Structure for Text and Image Encryption Using DNA Coding*, vol. 40, no. 9. Springer US, 2021.

[135] B. Akiwate and L. Parthiban, "A DNA Cryptographic Solution for Secured Image and Text Encryption," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 2, pp. 397–407, 2021.

[136] A. Kadhim and R. S. Ali, "Enhancement AES based on 3D chaos theory and DNA operations addition," *Karbala Int. J. Mod. Sci.*, vol. 5, no. 2, 2019.

[137] A. Belazi, M. Talha, S. Kharbech, and W. Xiang, "Novel Medical Image Encryption Scheme Based on Chaos and DNA Encoding," *IEEE Access*, vol. 7, pp. 36667–36681, 2019.

[138] A. Jain and N. Rajpal, "A robust image encryption algorithm resistant to attacks using DNA and chaotic logistic maps," *Multimed. Tools Appl.*, vol. 75, no. 10, pp. 5455–5472, 2016.

[139] J. Juremi, R. Mahmod, and S. Sulaiman, "A proposal for improving AES S-box with rotation and key-dependent," *Proc. 2012 Int. Conf. Cyber Secur. Cyber Warf. Digit. Forensic, CyberSec 2012*, pp. 38–42, 2012.

[140] T. Ara, P. G. Shah, and M. Prabhakar, "Dynamic key Dependent S-Box for Symmetric Encryption for IoT Devices," *Proc. 2018 2nd Int. Conf. Adv. Electron. Comput. Commun. ICAECC 2018*, pp. 1–5, 2018.

[141] B. Maram and J. M. Gnanasekar, "A Block Cipher Algorithm to Enhance the Avalanche Effect Using Dynamic Key-Dependent S-Box and Genetic Operations," *Int. J. Pure Appl. Math.*, vol. 119, no. 10, pp. 399–418, 2018.

[142] K. Kazlauskas, G. Vaicekauskas, and R. Smaliukas, “An Algorithm for Key-Dependent S-Box Generation in Block Cipher System,” *Inform.*, vol. 26, no. 1, pp. 51–65, 2015.

[143] M. Hintze and G. LaFever, “Meeting Upcoming GDPR Requirements While Maximizing the Full Value of Data Analytics,” *SSRN Electron. J.*, 2017.

[144] G. Dodig-crnkovic, “Scientific Methods in Computer Science,” no. October, 2012.

[145] R. Elio, J. Hoover, I. Nikolaidis, M. Salavatipour, L. Stewart, and K. Wong, “About Computing Science Research Methodology.”

[146] A. Dresch, D. P. Lacerda, and J. A. V. Antunes, “Design science research,” in *Design science research*, Springer, 2015, pp. 67–102.

[147] A. Elragal and M. Haddara, “Design Science Research: Evaluation in the Lens of Big Data Analytics,” *Systems*, vol. 7, no. 2, p. 27, 2019.

[148] N. Manson, “Is operations research really research?,” *ORiON*, vol. 22, no. 2, pp. 155–180, 2006.

[149] B. Norgeot *et al.*, “Protected Health Information filter (Philter): Accurately and Securely De-identifying Free-text Clinical Notes,” *npj Digit. Med.*, vol. 3, no. 1, pp. 1–8, 2020.

[150] W. Yihan and L. Yongzhen, “Improved Design of des Algorithm Based on Symmetric Encryption Algorithm,” *Proc. 2021 IEEE Int. Conf. Power Electron. Comput. Appl. ICPECA 2021*, pp. 220–223, 2021.

[151] J. H. Anajemba, C. Iwendi, M. Mittal, and T. Yue, “Improved advance encryption standard with a privacy database structure for IoT nodes,” *Proc. - 2020 IEEE 9th Int. Conf. Commun. Syst. Netw. Technol. CSNT 2020*, pp. 201–206, 2020.

[152] A. Rukhin, J. Soto, and J. Nechvatal, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” *Nist Spec. Publ.*, vol. 22, no. April, pp. 1/1----G/1, 2010.

[153] P. S. Appelbaum, “Privacy in psychiatric treatment: Threats and responses,” *Am. J. Psychiatry*, vol. 159, no. 11, pp. 1809–1818, 2002.

[154] J. Grimson, W. Grimson, and W. Hasselbring, “The SI challenge in health care,” *Commun. ACM*, vol. 43, no. 6, pp. 48–55, 2000.

[155] A. Stubbs and Ö. Uzuner, “Annotating Longitudinal Clinical Narratives for De-identification: The 2014 i2b2/UTHealth Corpus,” *J. Biomed. Inform.*, vol. 58, pp. S20--S29, 2015.

[156] J. Gardner and L. Xiong, “HIDE : An Integrated System for Health Information DE-identification HIDE : An Integrated System for Health Information DE-identification ∗,” no. July 2008, 2014.

[157] A. Stubbs, C. Kotfila, and Ö. Uzuner, “Automated Systems for the De-identification of Longitudinal Clinical narratives: Overview of 2014 i2b2/UTHealth shared task Track 1,” *J. Biomed. Inform.*, vol. 58, pp. S11--S19, 2015.

[158] G. Ibáñez-Sanz *et al.*, “Statin Use and the Risk of Colorectal Cancer in a Population-based Electronic HealthRecords Study,” *Sci. Rep.*, vol. 9, no. 1, pp. 1–8, 2019.

[159] K. Lam, T. Parkin, C. Riggs, and K. Morgan, “Use of Free Text Clinical Records in

Identifying Syndromes and Analysing Health Data," *Vet. Rec.*, vol. 161, no. 16, pp. 547–551, 2007.

[160] "Primary care data for public health researche," *Medicines & Healthcare products Regulatory Agency*. [Online]. Available: https://cprd.com/primary-care-data-public-health-research. [Accessed: 23-Apr-2022].

[161] R. Leone *et al.*, "Identifying Adverse Drug Reactions Associated with Drug-drug Interactions: Data Mining of A spontaneous Reporting Database in Italy," *Drug Saf.*, vol. 33, no. 8, pp. 667–675, 2010.

[162] L. Baril *et al.*, "Risk of Spontaneous Abortion and Other Pregnancy Outcomes in 15-25 Year Old Women Exposed to Human Papillomavirus-16/18 AS04-Adjuvanted Vaccine in the United Kingdom," *Vaccine*, vol. 33, no. 48, pp. 6884–6891, 2015.

[163] D. Heredia-Ductram, M. Nunez-Del-Prado, and H. Alatrista-Salas, "Toward a Comparison of Classical and New Privacy Mechanism," *Entropy*, vol. 23, no. 4, 2021.

[164] T. Ahmed, M. M. Al Aziz, and N. Mohammed, "De-identification of Electronic Health Record using Neural Network," *Sci. Rep.*, vol. 10, no. 1, pp. 1–11, 2020.

[165] S. Editors and A. Editors, *The Design of Rijndael*, 2nd ed. New York: Springer-verlag.

[166] A. R. Tate *et al.*, "Exploiting the potential of large databases of electronic health records for research using rapid search algorithms and an intuitive query interface," *J. Am. Med. Informatics Assoc.*, vol. 21, no. 2, pp. 292–298, 2014.

[167] R. G. Brown, "DieHarder: A Gnu Public License Random Number Tester," *Duke Univ. Phys. Dep. Durham*, 2018.

[168] P. L'ecuyer and R. Simard, "TestU01: A C library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, 2007.

[169] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. Bifurc. Chaos*, vol. 16, no. 8, pp. 2129–2151, 2006.

[170] J. B. Alimpia, "An Enhanced Hash-based Message Authentication Code using BCrypt," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 6, no. 4, pp. 1429–1432, 2018.

[171] R. M. Marzan and A. M. Sison, "An enhanced key security of playfair cipher algorithm," *ACM Int. Conf. Proceeding Ser.*, vol. Part F1479, no. February, pp. 457–461, 2019.

[172] V. Roman'kov, "Two General Schemes of Algebraic Cryptography," *Groups, Complexity, Cryptol.*, vol. 10, no. 2, pp. 83–98, 2018.

[173] C. Blondeau, G. Leander, and K. Nyberg, "Differential-Linear Cryptanalysis Revisited," *J. Cryptol.*, vol. 30, no. 3, pp. 859–888, 2017.

[174] H. Aamot, C. D. Kohl, D. Richter, and P. Knaup-Gregori, "Pseudonymization of Patient Identifiers for Translational Research," *BMC Med. Inform. Decis. Mak.*, vol. 13, no. 1, pp. 1–15, 2013.

[175] T. Neubauer and J. Heurix, "A methodology for the Pseudonymization of Medical Data," *Int. J. Med. Inform.*, vol. 80, no. 3, pp. 190–204, 2011.

[176] P. Geetha, C. Naikodi, and S. L. N. Setty, *Design of big data privacy framework—a balancing act*, vol. 612. Springer Singapore, 2020.

[177] J. Aberdeen *et al.*, “The MITRE Identification Scrubber Toolkit: Design, training, and assessment,” *Int. J. Med. Inform.*, vol. 79, no. 12, pp. 849–859, 2010.

[178] A. E. W. Johnson *et al.*, “MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports,” *Sci. Data*, vol. 6, no. 1, pp. 1–8, 2019.

[179] Z. Liu, B. Tang, X. Wang, and Q. Chen, “De-identification of clinical notes via recurrent neural network and conditional random field,” *J. Biomed. Inform.*, vol. 75, pp. S34--S42, 2017.

[180] J. L. Hernández-Ramos *et al.*, “Protecting Personal Data in IoT Platform Scenarios Through Encryption-based Selective Disclosure,” *Comput. Commun.*, vol. 130, no. July, pp. 20–37, 2018.

[181] Y. Wu, M. Jiang, J. Lei, and H. Xu, “Named Entity Recognition in Chinese Clinical Text Using Deep Neural Network.,” *Stud. Health Technol. Inform.*, vol. 216, pp. 624–8, 2015.

[182] M. Allahyari, E. D. Trippe, and J. B. Gutierrez, “A Brief Survey of Text Mining: Classification , Clustering and Extraction Techniques,” *arXiv*, 2017.

[183] S. Keretna, C. P. Lim, and D. Creighton, “A hybrid Model for Named Entity Recognition using Unstructured Medical Text,” *Proc. 9th Int. Conf. Syst. Syst. Eng. Socio-Technical Perspect. SoSE 2014*, pp. 85–90, 2014.

[184] A. Mishra and S. K. Jain, “A survey on Question Answering Systems with Classification,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 28, no. 3, pp. 345–361, 2016.

[185] K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao, “Question Answering on Freebase via Relation Extraction and Textual Evidence,” 2016.

[186] F. Dugas and E. Nichols, “DeepNNNER : Applying BLSTM-CNNs and Extended Lexicons to Named Entity Recognition in Tweets,” pp. 178–187, 2016.

[187] L. Derczynski *et al.*, “Analysis of Named Entity Recognition and Linking for Tweets,” *Inf. Process. Manag.*, vol. 51, no. 2, pp. 32–49, 2015.

[188] Q. Shi and M. Abdel-Aty, “Big Data applications in real-time traffic operation and safety monitoring and improvement on urban expressways,” *Transp. Res. Part C Emerg. Technol.*, vol. 58, pp. 380–394, 2015.

[189] T. Baldwin, M.-C. de Marneffe, B. Han, Y.-B. Kim, A. Ritter, and W. Xu, “Shared Tasks of the 2015 Workshop on Noisy User-generated Text: Twitter Lexical Normalization and Named Entity Recognition,” pp. 126–135, 2015.

[190] A. McCallum and W. Li, “Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons,” *Comput. Sci. Dep. Fac. Publ. Ser.*, 2003.

[191] S. M. Thomas, B. Mamlin, G. Schadow, and C. McDonald, “A successful Technique for Removing Names in Pathology Reports using an Augmented Search and Replace Method.,” *Proceedings. AMIA Symp.*, pp. 777–81, 2002.

[192] H. Gali, A. Surana, P. Vaidya, Shishtla, and D. M.Sharma, “Aggregating Machine Learning and Rule Based Heuristic for Named EntityRecognition,” *Proc. IJCNLP-08 Work. NER South South East Asian Lang.*, no. January, pp. 25–32, 2008.

[193] O. Ferrández, B. R. South, S. Shen, F. J. Friedlin, M. H. Samore, and S. M. Meystre, "Evaluating current automatic de-identification methods with Veterans health administration clinical documents," *BMC Med. Res. Methodol.*, vol. 12, 2012.

[194] A. Coden, D. Gruhl, N. Lewis, M. Tanenblatt, and J. Terdiman, "SPOT the Drug! An Unsupervised Pattern Matching Method to Extract Drug Names from Very large Clinical Corpora," *Proc. - 2012 IEEE 2nd Conf. Healthc. Informatics, Imaging Syst. Biol. HISB 2012*, pp. 33–39, 2012.

[195] B. A. Beckwith, R. Mahaadevan, U. J. Balis, and F. Kuo, "Development and evaluation of an open source software tool for deidentification of pathology reports," *BMC Med. Inform. Decis. Mak.*, vol. 6, pp. 1–10, 2006.

[196] A. Ekbal, S. Saha, and P. Bhattacharyya, "Deep Learning Architecture for Patient Data De-identification in Clinical Records," *Proc. Clin. Nat. Lang. Process. Work.*, pp. 32–41, 2016.

[197] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," pp. 1–38, 2015.

[198] S. Hochreiter and J. Urgen Schmidhuber, "Long Shortterm Memory," *Neural Comput.*, vol. 9, no. 8, p. 17351780, 1997.

[199] D. Nouvel, M. Ehrmann, and S. Rosset, "Evaluating Named Entity Recognition," in *Named Entities for Computational Linguistics*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2016, pp. 111–129.

[200] F. Dernoncourt, J. Y. Lee, O. Uzuner, and P. Szolovits, "De-identification of Patient Notes with Recurrent Neural Networks," *J. Am. Med. Informatics Assoc.*, vol. 24, no. December 2016, pp. 596–606, 2017.

[201] Z. Liu, B. Tang, X. Wang, and Q. Chen, "De-identification of Clinical Notes via Recurrent Neural Network and Conditional Random Field," *J. Biomed. Inform.*, vol. 75, pp. S34--S42, 2017.

[202] A. J. McMurry, B. Fitch, G. Savova, I. S. Kohane, and B. Y. Reis, "Improved De-identification of Physician Notes Through Integrative Modeling of both Public and Private Medical Text," *BMC Med. Inform. Decis. Mak.*, vol. 13, no. 1, 2013.

[203] M. Scaiano *et al.*, "A unified Framework for Evaluating the Risk of Re-identification of Text De-identification Tools," *J. Biomed. Inform.*, 2016.

[204] B. He, Y. Guan, J. Cheng, K. Cen, and W. Hua, "CRFs based de-identification of medical records," *J. Biomed. Inform.*, vol. 58, pp. S39–S46, 2015.

[205] B. Alabdullah, N. Beloff, and M. White, *ARTPHIL: Reversible De-identification of Free Text Using an Integrated Model*, no. March. Springer International Publishing, 2022.

[206] N. R. Al-Kazaz and W. J. Teahan, "An Automatic Cryptanalysis of Arabic Transposition Ciphers using Compression," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 11, pp. 738–745, 2018.

[207] S. Agustini, W. M. Rahmawati, and M. Kurniawan, "Modified Vegenere Cipher to Enhance Data Security Using Monoalphabetic Cipher," *Int. J. Artif. Intell. Robot.*, vol. 1, no. 1, p. 25, 2019.

[208] Amalia, M. A. Budiman, and R. Sitepu, “File Text Security using Hybrid Cryptosystem with Playfair Cipher Algorithm and Knapsack Naccache-Stern Algorithm,” *J. Phys. Conf. Ser.*, vol. 978, no. 1, 2018.

[209] R. M. Marzan, A. M. Sison, and R. P. Medina, “An Enhanced Key Security of Playfair Cipher Algorithm,” *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 4, pp. 1248–1253, 2019.

[210] T. M. Aung and N. N. Hla, “A Complex Polyalphabetic Cipher Technique Myanmar Polyalphabetic Cipher,” *2019 Int. Conf. Comput. Commun. Informatics, ICCCI 2019*, 2019.

[211] A. Elmogy, Y. Bouteraa, R. Alshabanat, and W. Alghaslan, “A New Cryptography Algorithm Based on ASCII Code,” *19th Int. Conf. Sci. Tech. Autom. Control Comput. Eng. STA 2019*, pp. 626–631, 2019.

[212] N. Yadav, R. K. Kapoor, and M. A. Rizvi, “A Novel Symmetric Key Cryptography using Dynamic Matrix Approach,” *Adv. Intell. Syst. Comput.*, vol. 439, pp. 51–60, 2016.

[213] A. Biryukov, *Encyclopedia of Cryptography and Security*. Boston, MA: Springer US, 2005.

[214] A. Sciences, *Springer Encyclopedia of Cryptography and Security*, no. March. 2016.

[215] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, vol. 8, no. 11. 1964.

[216] T. Nie and T. Zhang, “A study of DES and Blowfish Encryption Algorithm,” *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, pp. 1–4, 2009.

[217] N. Aleisa, “A Comparison of the 3DES and AES Encryption Standards,” *Int. J. Secur. its Appl.*, vol. 9, no. 7, pp. 241–246, 2015.

[218] V. R. Joan Daemen, “AES Proposal: Rijndael Joan,” pp. ix–xii, 2003.

[219] S. Manku and K. Vasanth, “Blowfish Encryption Algorithm for Information Security,” *ARPN J. Eng. Appl. Sci.*, vol. 10, no. 10, pp. 4717–4719, 2015.

[220] C. S. Lamba, “Design and Analysis of Stream Cipher for Network Security,” in *2010 Second International Conference on Communication Software and Networks*, 2010, vol. 76, pp. 526–567.

[221] S. Rajesh, V. Paul, V. G. Menon, and M. R. Khosravi, “A Secure and Efficient Lightweight Symmetric Encryption Scheme for Transfer of Text Files Between Embedded IoT Devices,” *Symmetry (Basel).*, vol. 11, no. 2, 2019.

[222] B. Rothke, “A look at the Advanced Encryption Standard (AES),” in *Information Security Management Handbook, Sixth Edition*, 2007, pp. 1151–1158.

[223] S. Arrag, “Design and Implementation A different Architectures of mixcolumn in FPGA,” *Int. J. VLSI Des. Commun. Syst.*, vol. 3, no. 4, pp. 11–22, 2012.

[224] N. B. Abdulwahed, “Chaos-Based Advanced Encryption Standard,” *Master Diss.*, 2013.

[225] and C. D. C. Biryukov, Alex, “DATA ENCRYPTION STANDARD (DES),” *Encycl. Cryptogr. Secur.*, pp. 295–301, 2011.

[226] B. De Decker, “Introduction to Computer Security,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 1528, pp. 377–393, 1998.

Appendices

Appendix I: E-ART Code (Java)

Below is the java source code for E-ART encryption and decryption algorithms developed in Chapter 4

```
/*
 * This code created to test the encryption and decryption of E-ART algorithm. it read document from the
  local path then encrypt it and upload it as new encrypted document. then it read the encrypt document and decrypt itand upload it
  as new decrypted document
 */
package eart;
import org.apache.commons.io.FileUtils;
import org.apache.commons.io.IOUtils;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Arrays;
import java.io.File;
import java.nio.charset.StandardCharsets;
import java.util.Random;

/**
 *
 * @author bayan
 */
public class Eart {

    // declarition of variables
    private boolean find = false;
    private int Low = 0;
    private int High = 9;
```

```
    private int max_length = 127;
    private int delta = 32;
    int pos_offset = 3;
    int N = 80000000;
    // declartion of string of array that will be used to represent the escap secuence charecters in the ascii table between 33 and 47.
    private String[] str_const = {"+!+", "", "+#+", "+$+", "+%+", "+&+", "+'+", "+yy+", "+zz+", "+*+", "+++", "+"+", "+ww+", "+.+",
"+xx+"};


/**this function read the text document, get the Variable offset (var-offset) and  dynamic offset (psudo)
     * and then encrypt each characters and append to the encrypt list and generate the encrypted file
     * @param localPath
     */
    public void Encrypt_data_art(String localPath) {
        try {
            File myfile = new File(localPath);
            // read file
            String str = FileUtils.readFileToString(myfile, StandardCharsets.UTF_8); // read file
            // convert the text document into array of string and Separate each character
            String[] string_array = str.split("");
             System.out.println(" E-ART encrption in progress");
            // Declared an  array of integer and  store the result of getVarOffset() function on it
            int [] ans = getVarOffset();
            // an offset that will be used in replaceCharAt () function
            int var_offset = ans[0];
            // an offset that will be used in replaceCharAt () function
            int const_offset = ans[1];
            // create instance of Random class
            Random r = new Random();
            // This loop ietrated over the the array of string, send the charecter postition i  and random integer r to create
            // pesudo random nubmer for each charecters in the document
            long pseudo = 0;
            for (int i = 0; i < string_array.length; i++) {
                pseudo = pseudorandom(i, r); //call function
            // send all the offsets needed to encrypt single charecter
                string_array[i] = replaceCharAt(string_array[i], pseudo, var_offset,const_offset );
            }
        // Once all characters in the plaintext are encrypted, the encrypted text file is generated.
            FileUtils.writeStringToFile(new File("C://test/encrypted.txt"), convertArrayToStringMethod(string_array), "UTF-8");
            System.out.println(" E-ART encrption is completed");

            decrypt_art (string_array); // call the dcryption function

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
     /**
     * @param value which represent the character from the array of string
     * @param index which represent the The pseudo-random number
     * @param var_offset
     * @param const_offset
     * @return str which represent the encrypted character
     */
    private String replaceCharAt(String value, Long index, int var_offset, int const_offset) {
        int org_value = value.charAt(0);// the  character  is converted into its corresponding ASCII value and stored in org_value.
        int reflected_value = (max_length - org_value) + 1; // generate the initial reflected value by substracting the org_value from the
max_lenth
        int lap_off = (int) (index % pos_offset); // The pseudo-random number is then adjusted using the pos_offset which is part of the
seceret key
       // create a StringBuilder object
       // usind StringBuilder() constructor
        String str = "";
        StringBuilder sb = new StringBuilder();
        if ((reflected_value + var_offset + const_offset) > max_length) {
            // avoid unpreintable charecters from 0 to 32 by adding delta
            if (((reflected_value + var_offset + const_offset) % max_length) < 32) {
                reflected_value = ((reflected_value + var_offset + const_offset) % max_length) + delta + lap_off;
                // avoid the are escape-sequence characters that cannot be printed. chaecters between 33 and 47.
                if (reflected_value >= 33 && reflected_value <= 47) {
```

```
                str = str_const[reflected_value - delta - 1];
            } else {
                str = sb.append((char) (reflected_value)).toString(); // convert the ascii number to the equlivant charecters
            }
        } else {
            reflected_value = ((reflected_value + var_offset + const_offset) % max_length) + lap_off;
            str = sb.append((char) (reflected_value)).toString();
        }
    } else {
        reflected_value = reflected_value + var_offset + const_offset + lap_off;// reflected value is converted to is converted to the
equivalent ASCII character to produce the encrypted character.
        str = sb.append((char) (reflected_value)).toString(); // append it to decrypt list
    }
    return str;
}

/**
 * @param strArray
 * @return  string
 */
public static String convertArrayToStringMethod(String[] strArray) {

    StringBuilder stringBuilder = new StringBuilder();

    for (int i = 0; i < strArray.length; i++) {

        stringBuilder.append(strArray[i]);
        //     stringBuilder.append(",");
    }

    return stringBuilder.toString();
}
/**
 * This function use  uses each character's position in the text as a seed to generate a pseudo-random number
 * @param seed which is the character position within the document
 * @param r which is random number
 * @return pseudorandom number
 */
public long pseudorandom(int seed, Random r) {
    // setting seed
    //  long s = 24;
    r.setSeed(seed);
    // value after setting seed
    // return Math.abs(r.nextInt());
    return Math.abs(r.nextLong());

}
/**
 * This function used to generate the var_offset using N, the first value of the initial key, and the properties of the tree—R, NL, and
NR
 * the constant_offset is modified based on the value of Var_offset to keep characters within the range
 * @return an array that contain var_offset and constant offset
 */

public  int[] getVarOffset()
{
    int ans [] = new int [2];
    //int N = 80000000;
    int sub = 0;
    int var_offset1;
    int const_offset1= 32;
    int offset_L = N % max_length;
    int offset_R = (max_length - offset_L)+1;
    int root = max_length /2 ;
    if (offset_L < root)

        var_offset1 =  (root * offset_L) % offset_R;

        else
```

```
            var_offset1 =  (root * offset_R) % offset_L;
    // keep the var_offset within the range
    if (var_offset1> 64){
        var_offset1 = (var_offset1% 64);}
    // modify the contant offset to keep the charecters within the range
    // var-offset have to be less than 39, otherwise the constant offset have to bre modify
    if (var_offset1 > 39){
       sub = var_offset1 - const_offset1;
       const_offset1 = const_offset1 + sub;}
     // modify the contant offset to keep the charecters within the range
    // var-offset have to be more than 31, otherwise the constant offset have to bre modify
    else if (var_offset1 < 31){
        //System.out.println ("var offset is > 31");
        sub = const_offset1 - var_offset1;
        const_offset1 = const_offset1 + sub;}
    ans [0] = var_offset1;
    ans [1] = const_offset1;
    return ans;

  }

/**
 *this function read the encrypted document, get the Variable offset (var-offset) and  dynamic offset (psudo)
 * and then decrypt each characters and append to the decrypt list and generate the decrypted file
 * @param string_array
 */

public void decrypt_art(String[] string_array ) {

    try {
        System.out.println(" E-ART decrption in progress");
        StringBuilder stringBuilder = new StringBuilder();
        // create instance of Random class
        Random r = new Random();
        long pseudo = 0;
        // This loop ietrated over the the array of string, send the charecter postition i  and random integer r to create
        // pesudo random nubmer for each charecters in the document
        for (int i = 0; i < string_array.length; i++) {
            //   System.out.println(string_array[i]);
            pseudo = pseudorandom(i, r);
            stringBuilder.append(deidentify(string_array[i], pseudo ));
        }

     // convert it to string
        String str_converted = stringBuilder.toString();
    // remove all extra charecters
        str_converted = str_converted.replace("@", " ");
        str_converted = str_converted.replace("*", " ");

        System.out.println(str_converted);
        // The decrypted text file is generated.
        System.out.println(" E-ART decrption is completed");
        FileUtils.writeStringToFile(new File("C://test/decrypted.txt"), str_converted, "UTF-8");
    } catch (Exception e) {
        e.printStackTrace();
    }
    //  disconnect();
}

/**
 * this function take an an encrypted character and the psudo offset, it adject the psudo offset and check if the characters
 * from the str array and then send the character to be decrypted by replaceChar(int_value_anonymized) function
 * then append the decrypted to the decrypted list and generate the decrypted file
 *
 * @param word which represent the encrypted character
 * @param index which represent the pusedo offset
 * @return
 */
public char deidentify(String word, long index ) {
```

```
        int int_value_anonymized = 0;
        char org_word = ' ';
        int lap_off = (int) (index % pos_offset);// The pseudo-random number is  adjusted using the pos_offset which is part of the seceret
k
        if (word.length() > 1) {
            word = word.substring(1, word.length() - 1);

            // check if the word is from the the  str_consts array that represent the escap charecters range of 33 - 47
            if (word.length() == 2) {
                if (word.equals("xx")) {
                    int_value_anonymized = 47 - delta - lap_off;
                    org_word = replaceChar(int_value_anonymized);
                } else if (word.equals("ww")) {
                    int_value_anonymized = 45 - delta - lap_off;
                    org_word = replaceChar(int_value_anonymized);
                } else if (word.equals("yy")) {
                    int_value_anonymized = 40 - delta - lap_off;
                    org_word = replaceChar(int_value_anonymized);
                } else if (word.equals("zz")) {
                    int_value_anonymized = 41 - delta - lap_off;
                    org_word = replaceChar(int_value_anonymized);
                } else if (word.equals("''")) {
                    int_value_anonymized = 44 - delta - lap_off;
                    org_word = replaceChar(int_value_anonymized);
                } else if (word.equals("uu")) {
                    org_word = 'D';
                }
            } else {
//  if the charecters is not from the str array then the charecters converted into its corresponding ASCII value and stored in
value_anonymized
                int_value_anonymized = (int) (word.charAt(0)) - delta - lap_off;
                // call replacechar function with prammeter value of anonymized to complete the decryption process
                org_word = replaceChar(int_value_anonymized);
            }

        } else if (word.length() > 0) {
            int_value_anonymized = (int) (word.charAt(0)) - lap_off;
// convert the charecter to its its corresponding ASCII value and subtract the lap_off which represent the dynamic offset and then
stored it in value_anonymized
            org_word = replaceChar(int_value_anonymized); //send the charecter for decyption
        }

        org_word = replaceChar(int_value_anonymized);//  send the charecter for decyption
        return org_word;
    }

    /**
     * this function take the encrypted character and decrypted by reversing the encryption process and using the same prameters
     * which constant offset psudo offset and variable offset
     *
     * @param int_value_anonymized which is the Ascii value of the decrypted character
     * @return
     */
    private char replaceChar(int int_value_anonymized) {

        // the var_offset and contant offset are genenrated using getVaroffset() function and stored in array ans[]
        int [] ans = getVarOffset();
         int var_offset = ans[0];
          int const_offset = ans[1];
   // use quotient to determain weather to multply by max_length or not ( reverse the mod operation)
        int quotient = 0;
         if ((int_value_anonymized - (const_offset + var_offset)) <= 0) {
            quotient = 1; // multiply by max_lengh
        } else {
            quotient = 0; // not multiply it by max_lengh
        }

        int_value_anonymized = (max_length * quotient + int_value_anonymized) - const_offset - var_offset;
        int org_value = (max_length - int_value_anonymized) + 1;
```

```
        if (org_value == 44) { // ???
            org_value = org_value + 14;
        }
       // the value was from the unprintable charechters so we subtract delta as to reverse the process
        if (org_value >= 0 && org_value <= 32) {
            org_value = org_value + delta;
        }

        return (char) (org_value); // Decrypted value is converted to the equivalent ASCII character to produce the decrypted character
    }


    public static void main(String[] args) {
        String localPath = "C://test/";

        Eart ftp = new Eart ();
        ftp.Encrypt_data_art(localPath + "plaintext.txt");

    }

}
```

Appendix II: E-ART Evaluation

Below is the Java source code for evaluating the performance of the proposed encryption algorithm E-ART against AES and DES that is used in chapter 5. The code is developed to measure the effect of the changing data size for each cryptography algorithm. It measures the processing time and memory consumption for different sizes of textual files ranging from 200 to 2000 KB. The processing time is considered the time that an encryption algorithm takes to produce a ciphertext from a plaintext. It is computed by using System.currentTimeMillis() to get the start time and the end time and calculate the difference.

Memory consumption is the amount of memory consumed during the encryption or decryption processes. It is computed as follow; the memory used before your code execution is calculated. Next, the memory used after your code execution is calculated and then the difference is calculated to get the actual memory consumption.

```
/*
 * This code was created to evaluate the performance of E-ART against AES and DES in terms of execution time and memory consumption.
 */
package evaluation;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
```

```
import java.io.File;
import java.io.FileWriter;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.io.InputStream;
import java.io.ByteArrayInputStream;
import java.nio.charset.StandardCharsets;
import com.jcraft.jsch.Channel;
import com.jcraft.jsch.ChannelSftp;
import com.jcraft.jsch.JSch;
import com.jcraft.jsch.JSchException;
import com.jcraft.jsch.Session;
import java.util.*;
import org.apache.commons.io.FileUtils;
import org.apache.commons.io.IOUtils;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;   // THis Base64 will be enabled for AES
import java.math.BigInteger;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import java.security.MessageDigest;
import java.util.Arrays;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

//import org.apache.commons.codec.binary.Base64; // ENabled for DES

//import org.apache.commons.codec.binary.Base64;

public class Evaluation {

   private boolean find = false;
   private int max_length = 127;
   private int delta = 32;
   int pos_offset = 3;
   int N = 80000000;
   // declartion of string of array that will be used to represent the escap secuence charecters in the ascii table between
33 and 47.
   private String[] str_const = {"+!+", "", "+#+", "+$+", "+%+", "+&+", "+'+", "+yy+", "+zz+", "+*+", "+++", "+"+",
"+ww+", "+.+", "+xx+"};
   private static SecretKeySpec secretKey;
   private static byte[] key;
   static long MEGABYTE = 1024L * 1024L;

/**this function read the text document, get the Variable offset (var-offset) and  dynamic offset (psudo)
    * and then encrypt each characters and append to the encrypt list and generate the encrypted file
    * @param localPath
    */
   public void Encrypt_data_art(String localPath) {
      try {
```

```
            File myfile = new File(localPath);
            // read file
            String str = FileUtils.readFileToString(myfile, StandardCharsets.UTF_8); // read file
            // convert the text document into array of string and Separate each character
            String[] string_array = str.split("");
             System.out.println(" E-ART encrption in progress");
            // Declared an  array of integer and  store the result of getVarOffset() function on it
            int [] ans = getVarOffset();
            // an offset that will be used in replaceCharAt () function
            int var_offset = ans[0];
            // an offset that will be used in replaceCharAt () function
            int const_offset = ans[1];
            // create instance of Random class
            Random r = new Random();
            // This loop ietrated over the the array of string, send the charecter postition i  and random integer r to create
            // pesudo random nubmer for each charecters in the document
            long pseudo = 0;
            for (int i = 0; i < string_array.length; i++) {
                pseudo = pseudorandom(i, r); //call function
                  // send all the offsets needed to encrypt single charecter
                string_array[i] = replaceCharAt(string_array[i], pseudo, var_offset,const_offset );
            }
          // Once all characters in the plaintext are encrypted, the encrypted text file is generated.
            FileUtils.writeStringToFile(new    File("C://test/encrypted.txt"),    convertArrayToStringMethod(string_array),
"UTF-8");
            System.out.println(" E-ART encrption is completed");

            //decrypt_art (string_array); // call the dcryption function
          } catch (Exception e) {
            e.printStackTrace();
          }
       }
        /**
        * @param value which represent the character from the array of string
        * @param index which represent the The pseudo-random number
        * @param var_offset
        * @param const_offset
        * @return str which represent the encrypted character
        */
       private String replaceCharAt(String value, Long index, int var_offset, int const_offset) {

          int org_value = value.charAt(0);// the  character  is converted into its corresponding ASCII value and stored in
org_value.
          int reflected_value = (max_length - org_value) + 1; // generate the initial reflected value by substracting the
org_value from the max_lenth
          int lap_off = (int) (index % pos_offset); // The pseudo-random number is then adjusted using the pos_offset which
is part of the seceret key
         // create a StringBuilder object
         // usind StringBuilder() constructor
          String str = "";
          StringBuilder sb = new StringBuilder();

          if ((reflected_value + var_offset + const_offset) > max_length) {
            // avoid unpreintable charecters from 0 to 32 by adding delta
            if (((reflected_value + var_offset + const_offset) % max_length) < 32) {
              reflected_value = ((reflected_value + var_offset + const_offset) % max_length) + delta + lap_off;
              // avoid the are escape-sequence characters that cannot be printed. chaecters between 33 and 47.
              if (reflected_value >= 33 && reflected_value <= 47) {
                str = str_const[reflected_value - delta - 1];
              } else {
                str = sb.append((char) (reflected_value)).toString(); // convert the ascii number to the equlivant charecters
              }
```

```
            } else {
                reflected_value = ((reflected_value + var_offset + const_offset) % max_length) + lap_off;
                str = sb.append((char) (reflected_value)).toString();
            }
        } else {
            reflected_value = reflected_value + var_offset + const_offset + lap_off;// reflected value is converted to is converted to the equivalent ASCII character to produce the encrypted character.
            str = sb.append((char) (reflected_value)).toString(); // append it to decrypt list
        }
        return str;
    }
    /**
     * @param strArray
     * @return  string
     */
    public static String convertArrayToStringMethod(String[] strArray) {

        StringBuilder stringBuilder = new StringBuilder();

        for (int i = 0; i < strArray.length; i++) {

            stringBuilder.append(strArray[i]);
            //      stringBuilder.append(",");
        }

        return stringBuilder.toString();
    }

    /**
     * This function use  uses each character's position in the text as a seed to generate a pseudo-random number
     * @param seed which is the character position within the document
     * @param r which is random number
     * @return pseudorandom number
     */
    public long pseudorandom(int seed, Random r) {
        // setting seed
        //  long s = 24;
        r.setSeed(seed);
        // value after setting seed
        // return Math.abs(r.nextInt());
        return Math.abs(r.nextLong());

    }
    /**
     * This function used to generate the var_offset using N, the first value of the initial key, and the properties of the tree—R, NL, and NR
     * the constant_offset is modified based on the value of Var_offset to keep characters within the range
     * @return an array that contain var_offset and constant offset
     */

    public  int[] getVarOffset()
    {
        int ans [] = new int [2];
        //int N = 80000000;
        int sub = 0;
        int var_offset1;
        int const_offset1= 32;
        int offset_L = N % max_length;
        int offset_R = (max_length - offset_L)+1;
        int root = max_length /2 ;
```

```
   if (offset_L < root)

         var_offset1 =  (root * offset_L) % offset_R;

         else

         var_offset1 =  (root * offset_R) % offset_L;

   // keep the var_offset within the range
   if (var_offset1> 64){
      var_offset1 = (var_offset1% 64);}
   // modify the contant offset to keep the charecters within the range
   // var-offset have to be less than 39, otherwise the constant offset have to bre modify
   if (var_offset1 > 39){
     sub = var_offset1 - const_offset1;
     const_offset1 = const_offset1 + sub;}
    // modify the contant offset to keep the charecters within the range
   // var-offset have to be more than 31, otherwise the constant offset have to bre modify
   else if (var_offset1 < 31){
      //System.out.println ("var offset is > 31");
      sub = const_offset1 - var_offset1;
      const_offset1 = const_offset1 + sub;}

   ans [0] = var_offset1;
   ans [1] = const_offset1;
   return ans;

  }

/**
 *this function read the encrypted document, get the Variable offset (var-offset) and  dynamic offset (psudo)
 * and then decrypt each characters and append to the decrypt list and generate the decrypted file
 * @param string_array
 */

public void decrypt_art(String localPath ) {

   try {
      System.out.println(" E-ART decryption in progress");
      String str = FileUtils.readFileToString(new File(localPath), StandardCharsets.UTF_8.name());
      StringBuilder stringBuilder = new StringBuilder();
      String[] string_array = str.split(",");
      // create instance of Random class
      Random r = new Random();
      long pseudo = 0;
      // This loop ietrated over the the array of string, send the charecter postition i  and random integer r to create
      // pesudo random nubmer for each charecters in the document
      for (int i = 0; i < string_array.length; i++) {
         //   System.out.println(string_array[i]);
         pseudo = pseudorandom(i, r);
         stringBuilder.append(deidentify(string_array[i], pseudo ));
      }

    // convert it to string
      String str_converted = stringBuilder.toString();
   // remove all extra charecters
      str_converted = str_converted.replace("@", " ");
      str_converted = str_converted.replace("*", " ");

      System.out.println(str_converted);
      // The decrypted text file is generated.
      System.out.println(" E-ART decrption is completed");
```

```
        FileUtils.writeStringToFile(new File("C://test/decrypted.txt"), str_converted, "UTF-8");

      } catch (Exception e) {
        e.printStackTrace();
      }

   }

   /**
    * this function take an an encrypted character and the psudo offset, it adject the psudo offset and check if the characters
    * from the str array and then send the character to be decrypted by replaceChar(int_value_anonymized) function
    * then append the decrypted to the decrypted list and generate the decrypted file
    * @param word which represent the encrypted character
    * @param index which represent the pusedo offset
    * @return
    */
   public char deidentify(String word, long index ) {

      int int_value_anonymized = 0;
      char org_word = ' ';
      int lap_off = (int) (index % pos_offset);// The pseudo-random number is  adjusted using the pos_offset which is part of the seceret k

      if (word.length() > 1) {
         word = word.substring(1, word.length() - 1);

         // check if the word is from the the  str_consts array that represent the escap charecters range of 33 - 47
         if (word.length() == 2) {

            if (word.equals("xx")) {
               int_value_anonymized = 47 - delta - lap_off;
               org_word = replaceChar(int_value_anonymized);
            } else if (word.equals("ww")) {
               int_value_anonymized = 45 - delta - lap_off;
               org_word = replaceChar(int_value_anonymized);
            } else if (word.equals("yy")) {
               int_value_anonymized = 40 - delta - lap_off;
               org_word = replaceChar(int_value_anonymized);
            } else if (word.equals("zz")) {
               int_value_anonymized = 41 - delta - lap_off;
               org_word = replaceChar(int_value_anonymized);
            } else if (word.equals("'")) {
               int_value_anonymized = 44 - delta - lap_off;
               org_word = replaceChar(int_value_anonymized);
            } else if (word.equals("uu")) {
               org_word = 'D';
            }
         } else {
//  if the charecters is not from the str array then the charecters converted into its corresponding ASCII value and stored in value_anonymized
            int_value_anonymized = (int) (word.charAt(0)) - delta - lap_off;
            // call replacechar function with prammeter value of anonymized to complete the decryption process
            org_word = replaceChar(int_value_anonymized);
         }

      } else if (word.length() > 0) {
         int_value_anonymized = (int) (word.charAt(0)) - lap_off;
// convert the charecter to its its corresponding ASCII value and subtract the lap_off which represent the dynamic offset and then stored it in value_anonymized
         org_word = replaceChar(int_value_anonymized); //send the charecter for decyption
      }
```

```
        org_word = replaceChar(int_value_anonymized);//  send the charecter for decyption
        return org_word;
    }

    /**
     * this function take the encrypted character and decrypted by reversing the encryption process and using the same
prameters
     * which constant offset psudo offset and variable offset
     *
     * @param int_value_anonymized which is the Ascii value of the decrypted character
     * @return
     */
    private char replaceChar(int int_value_anonymized) {

        // the var_offset and contant offset are genenrated using getVaroffset() function and stored in array ans[]
        int [] ans = getVarOffset();
         int var_offset = ans[0];
          int const_offset = ans[1];

  // use quotient to determain weather to multply by max_length or not ( reverse the mod operation)
        int quotient = 0;

         if ((int_value_anonymized - (const_offset + var_offset)) <= 0) {
            quotient = 1; // multiply by max_lengh
        } else {
            quotient = 0; // not multiply it by max_lengh
        }

        int_value_anonymized = (max_length * quotient + int_value_anonymized) - const_offset - var_offset;
        int org_value = (max_length - int_value_anonymized) + 1;

        if (org_value == 44) { // ???
            org_value = org_value + 14;
        }
       // the value was from the unprintable charechters so we subtract delta as to reverse the process
        if (org_value >= 0 && org_value <= 32) {
            org_value = org_value + delta;
        }

        return (char) (org_value); // Decrypted value is converted to the equivalent ASCII character to produce the
decrypted character
    }
    /**
     * @param localPath which is the text file that contain the plaintext
     * @param secretKey
     * This function take the plaintext file  and encrypted using AES encryption algorithm and upload it to the given path
     */

    public void encrypt_AES(String localPath, String secretKey) {
        try {
            System.out.println ("encryption in progress");

            File myfile = new File(localPath);
            String str = FileUtils.readFileToString(myfile, StandardCharsets.UTF_8.name());// read text and convert it to
byte
            String encryptedString = encrypt(str, secretKey);
            System.out.println(encryptedString);
            FileUtils.writeStringToFile(new File("C:/test//sample_aes_enc.txt"), encryptedString, "UTF-8");

        } catch (Exception ex) {
            System.out.print(ex);
```

```
        }
    }
    /**
     * @param strToEncrypt
     * @param secret
     * @return encrypted string
     */
    public static String encrypt(String strToEncrypt, String secret) {
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            // Retruns encrypted value
              return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
            //return null;
        } catch (Exception e) {
            System.out.println("Error while encrypting: " + e.toString());
        }
        return null;
    }

    /* @param remoteFilePath
     * @param localPath
     * @param secretKey
     */

    /**
     *
     * @param localPath which represent the encrypted file
     * @param secretKey
     * This function take the encrypted file  and decrypted using AES decryption algorithm and upload it to the given
path
     */
    public void decrypt_AES( String localPath, String secretKey) {


        try {

            File file = new File(localPath);
            String str = FileUtils.readFileToString(file, StandardCharsets.UTF_8.name());// read the encryption text from
file


            String decryptedString = decrypt(str, secretKey);// call AES decryption function
            System.out.println (decryptedString);
            FileUtils.writeStringToFile(new File("C:/test/sample_aes_dec.txt"), decryptedString, "UTF-8");// write the
decryption text to file


        } catch (Exception e) {
            e.printStackTrace();
        }
        // disconnect();
    }

    /**
     *
     * @param strToDecrypt
     * @param secret
     * @return
     */
    public static String decrypt(String strToDecrypt, String secret) {
```

```
        try {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);


            return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
            //return null;
        } catch (Exception e) {
            System.out.println("Error while decrypting: " + e.toString());
        }
        return null;
    }

    public static void setKey(String myKey) {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");

        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    /*
     * @param localPath which represent the plaintext file
     * @param secretKey
     * This function take the plaintext file  and encrypted using AES encryption algorithm and upload it to the given path
     */
    public void Des_Encryption (String localPath, String secretKey) {

                try{
          File myfile = new File(localPath);
          String str = FileUtils.readFileToString(myfile, StandardCharsets.UTF_8.name());
         String encrypted_des = encrypt_des(str, secretKey);
        System.out.println(encrypted_des );
            FileUtils.writeStringToFile(new File("C:/test//sample_des_enc.txt"), encrypted_des, "UTF-8");

        }
             catch (Exception ex) {
            System.out.print(ex);
        }
}

    /**
     *
     * @param localPath
     * @param secretKey
     * This function take the encrypted file  and decrypted using DES decryption algorithm and upload it to the given
path
     */
    public void Des_decryption (String localPath, String secretKey) {

                try{
          File myfile = new File(localPath);
          String str = FileUtils.readFileToString(myfile, StandardCharsets.UTF_8.name());
```

```
        String encrypted_des = decrypt_des(str, secretKey);
        System.out.println(encrypted_des );
          FileUtils.writeStringToFile(new File("C:/test//sample_des_dec.txt"), encrypted_des, "UTF-8");

        }
          catch (Exception ex) {
          System.out.print(ex);
        }
}

    /**
     *
     * @param message
     * @param SECRET_KEY
     * @return
     */
    public String encrypt_des(String message, String SECRET_KEY) {
        try {
            final MessageDigest md = MessageDigest.getInstance("md5");
            final byte[] digestOfPassword = md.digest(SECRET_KEY.getBytes());
            final byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);

            for (int j = 0, k = 16; j < 8;) {
                keyBytes[k++] = keyBytes[j++];
            }

            final SecretKey key = new SecretKeySpec(keyBytes, "DESede");
            final IvParameterSpec iv = new IvParameterSpec(new byte[8]);
            final Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, key, iv);
            final byte[] plainTextBytes = message.getBytes();
            final byte[] cipherText = cipher.doFinal(plainTextBytes);
            return new String(Base64.encodeBase64(cipherText));

        } catch (Exception ex) {
            System.out.print(ex);
        }

        return null;

    }

    /**
     *
     * @param message
     * @param SECRET_KEY
     * @return decrypted string
     */
    public String decrypt_des(String message, String SECRET_KEY) {
        try {
            final MessageDigest md = MessageDigest.getInstance("md5");
            final byte[] digestOfPassword = md.digest(SECRET_KEY.getBytes());
            final byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);

            for (int j = 0, k = 16; j < 8;) {
                keyBytes[k++] = keyBytes[j++];
            }

            final SecretKey key = new SecretKeySpec(keyBytes, "DESede");
            final IvParameterSpec iv = new IvParameterSpec(new byte[8]);
            final Cipher decipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
            decipher.init(Cipher.DECRYPT_MODE, key, iv);
```

```
            final byte[] plainText = decipher.doFinal(Base64.decodeBase64(message.getBytes()));
            return new String(plainText, "UTF-8");

        } catch (Exception ex) {
            System.out.print(ex);
        }
        return null;
    }


    public static void main(String[] args) {

        String localPath = "C://test/";
        final String secretKey = "abc";

        long startTime = System.currentTimeMillis();
        long beforeUsedMem = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();
        Evaluation ftp = new Evaluation ();
        //ftp.Encrypt_data_art(localPath + "plaintext.txt");
        //ftp.decrypt_art(localPath + "encrypted.txt");
        ftp.encrypt_AES (localPath + "plaintext.txt", secretKey);
        //ftp.decrypt_AES (localPath + "sample_aes_enc.txt", secretKey);
        //ftp.Des_decryption (localPath + "sample_des_enc.txt", secretKey);
        //ftp.Des_Encryption(localPath + "plaintext.txt", secretKey);
        // claclulate the the time consumed for encyption
        long stopTime = System.currentTimeMillis();
        long elapsedTime = stopTime - startTime;
        System.out.println(" ");
        System.out.println("Time in milliseconds:--- " + elapsedTime);
        // calculate the the memory consumption
        long afterUsedMem = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();
        long actualMemUsed = afterUsedMem - beforeUsedMem;
        System.out.println("Memory consumed in Mega Bytes: --- " + actualMemUsed / MEGABYTE);
    }


}
```

Avalanche effect

```
import hashlib


def hamming_distance(chaine1, chaine2):
    return sum(c1 != c2 for c1, c2 in zip(chaine1, chaine2))


def hamming_distance2(chaine1, chaine2):
    return len(list(filter(lambda x : ord(x[0])^ord(x[1]), zip(chaine1, chaine2))))


if __name__=="__main__":
```

```
    fh = open( 'sample_des_enc1.txt' , encoding='utf-8')
    chaine1 = fh.read()
    fh.close()
#
    fh = open( 'sample_des_enc2.txt', encoding='utf-8' )
    chaine2 = fh.read()
    fh.close()

   # chaine1 = hashlib.md5("chaine1".encode()).hexdigest()
   # chaine2 = hashlib.md5("chaine2".encode()).hexdigest()

   # chaine1 = "6fb17381822a6ca9b02153d031d5d3da"
   # chaine2 = "a242eace2c57f7a16e8e872ed2f2287d"

    #chaine1="6e29201190152df4ee058139def610bb"
    #chaine2="b363bb16dd0a17b903ed4decba9223b8"

   # chaine1="c3b44b95d9d2f25670eee9a0de099fa3"
   # chaine2="4a901843a730165d716f4018074a11d5"

    res1 = ''.join(format(ord(i), 'b') for i in chaine1)
    res2 = ''.join(format(ord(i), 'b') for i in chaine2)


   # assert len(chaine1) == len(chaine2)

    print(hamming_distance(res1, res2)/len(res1) * 100)
    print(hamming_distance(res1, res2))
```

```
print ((len(chaine1)))
print ((len(res1)))
```

Appendix III: ARTPHIL (Python)

Below is the python source code for ARTPHIL de-identification system in Chapter (). It is a command line de-identification software that provide an end-to-end pipeline to detect and encrypt PHI from any plain text file. It is pull each text file from the directory, detect all PHI word and replace with encrypted string that can be regenerated and then write it to the output file with the same text file name appended with phi_reduced. It also generates additional output files containing meta data from the run: number of files processed, number of instances of PHI that were filtered, list of filtered words.

```
# De-id script
# import modules
from __future__ import print_function
import os
import sys
import pickle
import glob
import string
import re
import time
import argparse
# import multiprocess
import multiprocessing
from multiprocessing import Pool
# import NLP packages
import nltk
from nltk import sent_tokenize
from nltk import word_tokenize
from nltk.tree import Tree
from nltk import pos_tag_sents
from nltk import pos_tag
from nltk import ne_chunk
import spacy
from pkg_resources import resource_filename
from nltk.tag.perceptron import AveragedPerceptron
from nltk.tag import SennaTagger
from nltk.tag import HunposTagger
import array
import csv
import random


"""
detect PHI word and replace it with encrypted string
```

Does:
1. regex to search for salutations (must be done prior to splitting into sentences b/c of '.' present in most salutations)
2. split document into sentences.
3.Run regex patterns to identify PHI considering only 1 word at a time:emails, phone numbers, DOB, SSN, Postal codes, or any word containing 5 or more consecutive digits or
8 or more characters that begins and ends with digits.

4. Split sentences into words

5. Run regex patterns to identify PHI looking at the context for each word. For example DOB checks the preceding words for 'age' or 'years' etc.
addresses which include [streets, rooms, states, etc],age over 90.

6. Use nltk to label POS.

7. Identify names: We run 2 separate methods to check if a word is a name based on it's context at the chunk/phrase level. To do this:
First: Spacy nlp() is run on the sentence level and outputs NER labels at the chunk/phrase level.
Second: For chunks/phrases that spacy thinks are 'person', get a second opinion by running nltk ne_chunck which uses nltk POS
to assign an NER label to the chunk/phrases.
*If both spacy and nltk provide a 'person' NER label for a chunk/phrase: check the in the chunk 1-by-1 with nltk to determine if
the word's POS tag is a proper noun.
- sometimes the label 'person' may be applied to more than 1 word, and occassionally 1 of those words is just a normal noun, not a name.
- If word is a proper noun, flag the word and add it to name_set
*If spacy labels word as person but nltk does not label person but labels word as another category of NER, use spacy on the all UPPERCASE version words
in the chunk 1-by-1 to see if spacy still believes that the uppercase word is NER of any category
- If it is, add word to name_set;
- If spacy thinks the uppercase version of the word no longer has an NER label, then treat word as any other noun and send to be filtered through the whitelist.

8. If word is noun, send it on to check it against the whitelist. If word is not noun,
consider it safe and pass it on to output.For nouns, if word is in whitelist,
check if word is in name_set, if so -> filter.
If not in name_set,
use spacy to check if word is a name based on the single word's meaning and format.
Spacy does a per-word look up and assigns most frequent use of that word as a flag
(eg'HUNT':-organization, 'Hunt'-name, 'hunt':verb).
If the flags is name -> filter
If flag is not name pass word through as safe
if word not in whitelist -> filter
9. Search for middle initials by checking if single Uppercase letter is between PHI infos, if so, consider the letter as a middle initial and filter it. e.g. Ane H Berry.

NOTE: All of the above numbered steps happen in filter_task(). Other functions either support filter task or simply involve
dealing with I/O and multiprocessing

"""

```
#nlp = spacy.load('en')  # load spacy english library
nlp=spacy.load("en_core_web_sm")
# pretrain = SennaTagger('senna')

# configure the regex patterns
# we're going to want to remove all special characters
pattern_word = re.compile(r"[^\w+]")

# Find numbers like SSN/PHONE/FAX
```

```
# 3 patterns: 1. 6 or more digits will be filtered 2. digit followed by - followed by digit. 3. Ignore case of characters
pattern_number = re.compile(r"""\b(
(\d[\(\)\-\']?\s?){6}([\(\)\-\']?\d)+   # SSN/PHONE/FAX XXX-XX-XXXX, XXX-XXX-XXXX, XXX-XXXXXXXX, etc.
|(\d[\(\)\-.\']?){7}([\(\)\-.\']?\d)+  # test
)\b""", re.X)

pattern_4digits = re.compile(r"""\b(
\d{5}[A-Z0-9]*
)\b""", re.X)

pattern_devid = re.compile(r"""\b(
[A-Z0-9\-/]{6}[A-Z0-9\-/]*
)\b""", re.X)
# postal code
# 5 digits or, 5 digits followed dash and 4 digits
pattern_postal = re.compile(r"""\b(
\d{5}(-\d{4})?            # postal code XXXXX, XXXXX-XXXX
)\b""", re.X)

# match DOB
pattern_dob = re.compile(r"""\b(
.*?(?=\b(\d{1,2}[-./\s]\d{1,2}[-./\s]\d{2}  # X/X/XX
|\d{1,2}[-./\s]\d{1,2}[-./\s]\d{4}          # XX/XX/XXXX
|\d{2}[-./\s]\d{1,2}[-./\s]\d{1,2}          # xx/xx/xx
|\d{4}[-./\s]\d{1,2}[-./\s]\d{1,2}          # xxxx/xx/xx
)\b)
)\b""", re.X | re.I)

# match emails
pattern_email = re.compile(r"""\b(
[a-zA-Z0-9_.+-@\"]+@[a-zA-Z0-9-\:\]\[]+[a-zA-Z0-9-.]*
)\b""", re.X | re.I)

# match date, similar to DOB but does not include any words
month_name = "Jan(uary)?|Feb(ruary)?|Mar(ch)?|Apr(il)?|May|Jun(e)?|Jul(y)?|Aug(ust)?|Sep(tember)?|Oct(ober)?|Nov(ember)?|Dec(ember)?"
pattern_date = re.compile(r"""\b(
\d{4}[\-/](0?[1-9]|1[0-2]|"""+month_name+r""")\-\d{4}[\-/](0?[1-9]|1[0-2]|"""+month_name+r""")     #  YYYY/MM-YYYY/MM
|(0?[1-9]|1[0-2]|"""+month_name+r""")[\-/]\d{4}\-(0?[1-9]|1[0-2]|"""+month_name+r""")[\-/]\d{4}     #  MM/YYYY-MM/YYYY
|(0?[1-9]|1[0-2]|"""+month_name+r""")/\d{2}\-(0?[1-9]|1[0-2]|"""+month_name+r""")/\d{2}  # MM/YY-MM/YY
|(0?[1-9]|1[0-2]|"""+month_name+r""")/\d{2}\-(0?[1-9]|1[0-2]|"""+month_name+r""")/\d{4}          #     MM/YYYY-MM/YYYY
|(0?[1-9]|1[0-2]|"""+month_name+r""")/([1-2][0-9]|3[0-1]|0?[1-9])\-(0?[1-9]|1[0-2]|"""+month_name+r""")/([1-2][0-9]|3[0-1]|0?[1-9])  #MM/DD-MM/DD
|([1-2][0-9]|3[0-1]|0?[1-9])/(0?[1-9]|1[0-2]|"""+month_name+r""")\-([1-2][0-9]|3[0-1]|0?[1-9])/(0?[1-9]|1[0-2]|"""+month_name+r""")  #DD/MM-DD/MM
|(0?[1-9]|1[0-2]|"""+month_name+r""")[\-/\s]([1-2][0-9]|3[0-1]|0?[1-9])[\-/\s]\d{2}  # MM/DD/YY
|(0?[1-9]|1[0-2]|"""+month_name+r""")[\-/\s]([1-2][0-9]|3[0-1]|0?[1-9])[\-/\s]\d{4}  # MM/DD/YYYY
|([1-2][0-9]|3[0-1]|0?[1-9])[\-/\s](0?[1-9]|1[0-2]|"""+month_name+r""")[\-/\s]\d{2}  # DD/MM/YY
|([1-2][0-9]|3[0-1]|0?[1-9])[\-/\s](0?[1-9]|1[0-2]|"""+month_name+r""")[\-/\s]\d{4}  # DD/MM/YYYY
|\d{2}[\-./\s](0?[1-9]|1[0-2]|"""+month_name+r""")[\-\./\s]([1-2][0-9]|3[0-1]|0?[1-9])   # YY/MM/DD
|\d{4}[\-./\s](0?[1-9]|1[0-2]|"""+month_name+r""")[\-\./\s]([1-2][0-9]|3[0-1]|0?[1-9])   # YYYY/MM/DD

|\d{4}[\-/](0?[1-9]|1[0-2]|"""+month_name+r""")  # YYYY/MM
|(0?[1-9]|1[0-2]|"""+month_name+r""")[\-/]\d{4}  # MM/YYYY
|(0?[1-9]|1[0-2]|"""+month_name+r""")/\d{2}  # MM/YY
|(0?[1-9]|1[0-2]|"""+month_name+r""")/\d{2}  # MM/YYYY
|(0?[1-9]|1[0-2]|"""+month_name+r""")/([1-2][0-9]|3[0-1]|0?[1-9])  #MM/DD
```

```
|([1-2][0-9]|3[0-1]|0?[1-9])/(0?[1-9]|1[0-2]|"""+month_name+r""")  #DD/MM
)\b""", re.X | re.I)
pattern_mname = re.compile(r'\b(' + month_name + r')\b')

# match names, A'Bsfs, Absssfs, A-Bsfsfs
pattern_name = re.compile(r"""^[A-Z]\'?[-a-zA-Z]+$""")

# match age
pattern_age = re.compile(r"""\b(
age|year[s-]?\s?old|y.o[.]?
)\b""", re.X | re.I)

# match salutation
pattern_salutation = re.compile(r"""
(Dr\.|Mr\.|Mrs\.|Ms\.|Miss|Sir|Madam)\s
(([A-Z]\'?[A-Z]?[\-a-z]+(\s[A-Z]\'?[A-Z]?[\-a-z]+)*)
)""", re.X)

# match middle initial
# if single char or Jr is surround by 2 phi words, filter.
pattern_middle = re.compile(r"""\*\*PHI\*\*,? (([A-CE-LN-Z][Rr]?|[DM])\.?) | (([A-CE-LN-Z][Rr]?|[DM])\.?),?
\*\*PHI\*\*""")


# match url
pattern_url = re.compile(r'\b((http[s]?://)?([a-zA-Z0-9$-_@.&+:!\*\(\),])*[\.\/]([a-zA-Z0-9$-_@.&+:\!\*\(\),])*)\b',
re.I)

find = False
Low = 0;
High = 9;
max_length = 127;
delta = 32

#array('i');
ascii = array.array('i',(0 for i in range(0,max_length)))
#BinaryTree tree;
#const_offset = 32;

#var_offset = ((root_node * left_node) % right_node)
#var_offset = 32;


pos_offset = 3;


char_array = ["!","","#","$","%","&","","yy","zz","*","+","bb","ww",".","xx"]

ch_array = ["A","B","C","D","E","F","G","H","I","J"]
int_array= ["0","1","2","3","4","5","6","7","8","9"]

# check if the folder exists
def is_valid_file(parser, arg):
    if not os.path.exists(arg):
        parser.error("The folder %s does not exist. Please input a new folder or create one." % arg)
    else:
        return arg

def namecheck(word_output, name_set, screened_words, safe):
    # check if the word is in the name list
    if word_output.title() in name_set:
        # with open("name.txt", 'a') as fout:
```

```
            # fout.write(word_output + '\n')
        # print('Name:', word_output)
        screened_words.append(word_output)
      #  word_output = "**PHI**"
        safe = False

    else:
    # check spacy, and add the word to the name list if it is a name
    # check the word's title version and its uppercase version
        word_title = nlp(word_output.title())
        # search Title or UPPER version of word in the english dictionary: nlp()
        # nlp() returns the most likely NER tag (word.ents) for the word
        # If word_title has NER = person AND word_upper has ANY NER tag, filter
        word_upper = nlp(word_output.upper())
        if (word_title.ents != () and word_title.ents[0].label_ == 'PERSON' and
              word_upper.ents != () and word_upper.ents[0].label_ is not None):
            # with open("name.txt", 'a') as fout:
              # fout.write(word_output + '\n')
            # print('Name:', word_output)
            screened_words.append(word_output)
            name_set.add(word_output.title())
          #  word_output = "**PHI**"
            safe = False

    return word_output, name_set, screened_words, safe



def applyReflection(word, index):

        random.seed(index)
        posit = random.randint(1, 50) % pos_offset
       # print (posit)
        ascii_value = 0;
        str_reflected ='';
 #       var_offset = (root_node * left_node) % right_node
        length =  len(word);

        for pos in range(0, length):
            ascii_value = ord(word[pos]);
            if (posit > 0):
                lap_off = (index + pos) % (posit);
            else:
                lap_off = 0; #(index + pos)
            # lap_off = 0
            # print (ascii_value)
            str_reflected = replaceCharAt(str_reflected, ascii_value, lap_off);
        #word = 'YES'
    #    return str_reflected + append_index(index)+"|";'
        return str_reflected + "+"+append_index(index)+"+";


def append_index(index):

    output_string="";
    int_string = str(index)

    # Iterate over the string
    for element in int_string:
 #       print(ch_array[int(element)])
        output_string = output_string + ch_array[int(element)]
```

```
    return output_string


def replaceCharAt(str_reflected, org_value, lap_off):

        reflected_value = (max_length - org_value) + 1;
        const_offset, var_offset =  get_Varoffset();
        if ((reflected_value + var_offset + const_offset) > max_length):
              if (((reflected_value + var_offset + const_offset) % max_length) < 32):
                  reflected_value = ((reflected_value + var_offset + const_offset) % max_length)+ delta + lap_off;
                  if reflected_value >= 33 and reflected_value <= 47:
                 #     str_reflected = str_reflected + '+'+ char_array[reflected_value-delta-1] +'+|';
                      str_reflected = str_reflected + '+'+ char_array[reflected_value-delta-1] + '+';

                  else:
                     # str_reflected = str_reflected + '+'+ chr(reflected_value) +'+|';
                      str_reflected = str_reflected + '+'+ chr(reflected_value) + '+';

              else:
                  reflected_value = ((reflected_value + var_offset + const_offset) % max_length) + lap_off;
                #  str_reflected = str_reflected + chr(reflected_value) +'|';
                  str_reflected = str_reflected + chr(reflected_value);
        else:
            reflected_value = reflected_value + var_offset + const_offset + lap_off;
           # str_reflected = str_reflected + chr(reflected_value) +'|';
            str_reflected = str_reflected + chr(reflected_value);
        return str_reflected;

def get_Varoffset ():

    N = 70000000;
    sub = 0;
    #var_offset1;
    const_offset1= 32;
    offset_L = N % max_length;
    offset_R = (max_length - offset_L)+1;
    root = 65
    # root = (max_length +1) /2 ;
    var_offset1= 32

    if (offset_L < root):

      var_offset1 =  (root * offset_L) % offset_R;

    else:
     var_offset1 =  (root * offset_R) % offset_L;

       #keep the var_offset within the range
    if (var_offset1> 64):
      var_offset1 = (var_offset1% 64);
        # modify the contant offset to keep the charecters within the range
    #     # var-offset have to be less than 39, otherwise the constant offset have to bre modify
    if (var_offset1 > 39):
         sub = var_offset1 - const_offset1;
         const_offset1 = const_offset1 + sub;
         # modify the contant offset to keep the charecters within the range
    #     # var-offset have to be more than 31, otherwise the constant offset have to bre modify
    else:

     if (var_offset1 < 31):
      print ("var offset is > 31");
      sub = const_offset1 - var_offset1;
```

```
  const_offset1 = const_offset1 + sub;
return const_offset1, var_offset1


def filter_task(f, whitelist_dict, foutpath, key_name):

    # pretrain = HunposTagger('hunpos.model', 'hunpos-1.0-linux/hunpos-tag')
    pretrain = SennaTagger('senna')

    """
    Uses: namecheck() to check if word that has been tagged as name by either nltk or spacy. namecheck() first searches
    nameset which is generated by checking words at the sentence level and tagging names. If word is not in nameset,
    namecheck() uses spacy.nlp() to check if word is likely to be a name at the word level.

    """
    with open(f, encoding='utf-8', errors='ignore') as fin:
        # define intial variables
        head, tail = os.path.split(f)
        #f_name = re.findall(r'[\w\d]+', tail)[0]  # get the file number
        print(tail)
        start_time_single = time.time()
        total_records = 1
        phi_containing_records = 0
        safe = True
        screened_words = []
        name_set = set()
        phi_reduced = ''
        '''
        address_indictor = ['street', 'avenue', 'road', 'boulevard',
                            'drive', 'trail', 'way', 'lane', 'ave',
                            'blvd', 'st', 'rd', 'trl', 'wy', 'ln',
                            'court', 'ct', 'place', 'plc', 'terrace', 'ter']
                            '''
        address_indictor = ['street', 'avenue', 'road', 'boulevard',
                            'drive', 'trail', 'way', 'lane', 'ave',
                            'blvd', 'st', 'rd', 'trl', 'wy', 'ln',
                            'court', 'ct', 'place', 'plc', 'terrace', 'ter',
                            'highway', 'freeway', 'autoroute', 'autobahn', 'expressway',
                            'autostrasse', 'autostrada', 'byway', 'auto-estrada', 'motorway',
                            'avenue', 'boulevard', 'road', 'street', 'alley', 'bay', 'drive',
                            'gardens', 'gate', 'grove', 'heights', 'highlands', 'lane', 'mews',
                            'pathway', 'terrace', 'trail', 'vale', 'view', 'walk', 'way', 'close',
                            'court', 'place', 'cove', 'circle', 'crescent', 'square', 'loop', 'hill',
                            'causeway', 'canyon', 'parkway', 'esplanade', 'approach', 'parade', 'park',
                            'plaza', 'promenade', 'quay', 'bypass']


        note = fin.read()
        note = re.sub(r'=', ' = ', note)
        # Begin Step 1: saluation check
        re_list = pattern_salutation.findall(note)
        for i in re_list:
            name_set = name_set | set(i[1].split(' '))

        # note_length = len(word_tokenize(note))
        # Begin step 2: split document into sentences
        note = sent_tokenize(note)

        for sent in note: # Begin Step 3: Pattern checking
            # postal code check
            # print(sent)
            if pattern_postal.findall(sent) != []:
```

```
        safe = False
        for item in pattern_postal.findall(sent):
            screened_words.append(item[0])
    #  sent = str(pattern_postal.sub('**PHIPostal**', sent))

    if pattern_devid.findall(sent) != []:
        safe = False
        for item in pattern_devid.findall(sent):
            if (re.search(r'\d', item) is not None and
                re.search(r'[A-Z]',item) is not None):
                screened_words.append(item)
             #   sent = sent.replace(item, '**PHI**')

    # number check
    if pattern_number.findall(sent) != []:
        safe = False
        for item in pattern_number.findall(sent):
            # print(item)
            #if pattern_date.match(item[0]) is None:
          #  sent = sent.replace(item[0], '**PHI**')
            screened_words.append(item[0])
            #print(item[0])
    #sent = str(pattern_number.sub('**PHI**', sent))

    data_list = []
    if pattern_date.findall(sent) != []:
        safe = False
        for item in pattern_date.findall(sent):
            if '-' in item[0]:
                if (len(set(re.findall(r'[^\w\-]',item[0]))) <= 1):
                    #screened_words.append(item[0])
                    #print(item[0])
                    data_list.append(item[0])
                    #sent = sent.replace(item[0], '**PHIDate**')
            else:
                if len(set(re.findall(r'[^\w]',item[0]))) == 1:
                    #screened_words.append(item[0])
                    #print(item[0])
                    data_list.append(item[0])
                    #sent = sent.replace(item[0], '**PHIDate**')
    data_list.sort(key=len, reverse=True)
    #for item in data_list:
      # sent = sent.replace(item, '**PHIDate**')

    #sent = str(pattern_date.sub('**PHI**', sent))
    #print(sent)
    if pattern_4digits.findall(sent) != []:
        safe = False
        for item in pattern_4digits.findall(sent):
            screened_words.append(item)
#   sent = str(pattern_4digits.sub('**PHI**', sent))
    # email check
    if pattern_email.findall(sent) != []:
        safe = False
        for item in pattern_email.findall(sent):
            screened_words.append(item)
#   sent = str(pattern_email.sub('**PHI**', sent))
    # url check
    if pattern_url.findall(sent) != []:
        safe = False
        for item in pattern_url.findall(sent):
            #print(item[0])
```

```
        if (re.search(r'[a-z]', item[0]) is not None and
            '.' in item[0] and
            re.search(r'[A-Z]', item[0]) is None and
            len(item[0])>10):
            print(item[0])
            screened_words.append(item[0])
         #   sent = sent.replace(item[0], '**PHI**')
            #print(item[0])
#sent = str(pattern_url.sub('**PHI**', sent))
# dob check
'''
re_list = pattern_dob.findall(sent)
i = 0
while True:
    if i >= len(re_list):
        break
    else:
        text = ' '.join(re_list[i][0].split(' ')[-6:])
        if re.findall(r'\b(birth|dob)\b', text, re.I) != []:
            safe = False
            sent = sent.replace(re_list[i][1], '**PHI**')
            screened_words.append(re_list[i][1])
        i += 2
'''

# Begin Step 4
# substitute spaces for special characters
sent = re.sub(r'[\/\-\:\~\_]', ' ', sent)
# label all words for NER using the sentence level context.
spcy_sent_output = nlp(sent)
# split sentences into words
sent = [word_tokenize(sent)]
#print(sent)
# Begin Step 5: context level pattern matching with regex
for position in range(0, len(sent[0])):
    word = sent[0][position]
    # age check
    if word.isdigit() and int(word) > 90:
        if position <= 2:  # check the words before age
            word_previous = ' '.join(sent[0][:position])
        else:
            word_previous = ' '.join(sent[0][position - 2:position])
        if position >= len(sent[0]) - 2:  # check the words after age
            word_after = ' '.join(sent[0][position+1:])
        else:
            word_after = ' '.join(sent[0][position+1:position +3])

        age_string = str(word_previous) + str(word_after)
        if pattern_age.findall(age_string) != []:
            screened_words.append(sent[0][position])
         #   sent[0][position] = '**PHI**'
            safe = False

    # address check
    elif (position >= 1 and position < len(sent[0])-1 and
          (word.lower() in address_indictor or
           (word.lower() == 'dr' and sent[0][position+1] != '.')) and
          (word.istitle() or word.isupper())):

        if sent[0][position - 1].istitle() or sent[0][position-1].isupper():
            screened_words.append(sent[0][position - 1])
         #   sent[0][position - 1] = '**PHI**'
```

```
                i = position - 1
                # find the closet number, should be the number of street
                while True:
                    if re.findall(r'^[\d-]+$', sent[0][i]) != []:
                        begin_position = i
                        break
                    elif i == 0 or position - i > 5:
                        begin_position = position
                        break
                    else:
                        i -= 1
                i = position + 1
                # block the info of city, state, apt number, etc.
                while True:
                    if '**PHIPostal**' in sent[0][i]:
                        end_position = i
                        break
                    elif i == len(sent[0]) - 1:
                        end_position = position
                        break
                    else:
                        i += 1
                if end_position <= position:
                    end_position = position

                for i in range(begin_position, end_position):
                    #if sent[0][i] != '**PHIPostal**':
                    # screened_words.append(sent[0][i])
            #       sent[0][i] = '**PHI**'
                    safe = False

        # Begin Step 6: NLTK POS tagging
        sent_tag = nltk.pos_tag_sents(sent)
        #try:
            # senna cannot handle long sentence.
            #sent_tag = [[]]
            #length_100 = len(sent[0])//100
            #for j in range(0, length_100+1):
                #[sent_tag[0].append(j) for j in pretrain.tag(sent[0][100*j:100*(j+1)])]
            # hunpos needs to change the type from bytes to string
            #print(sent_tag[0])
            #sent_tag = [pretrain.tag(sent[0])]
            #for j in range(len(sent_tag[0])):
                #sent_tag[0][j] = list(sent_tag[0][j])
                #sent_tag[0][j][1] = sent_tag[0][j][1].decode('utf-8')
        #except:
            #print('POS error:', tail, sent[0])
            #sent_tag = nltk.pos_tag_sents(sent)
        # Begin Step 7: Use both NLTK and Spacy to check if the word is a name based on sentence level NER label
for the word.
        for ent in spcy_sent_output.ents:  # spcy_sent_output contains a dict with each word in the sentence and its NLP
labels
            #spcy_sent_ouput.ents is a list of dictionaries containing chunks of words (phrases) that spacy believes are
Named Entities
            # Each ent has 2 properties: text which is the raw word, and label_ which is the NER category for the word
            if ent.label_ == 'PERSON':
            #print(ent.text)
                # if word is person, recheck that spacy still thinks word is person at the word level
                spcy_chunk_output = nlp(ent.text)
                if spcy_chunk_output.ents != () and spcy_chunk_output.ents[0].label_ == 'PERSON':
                    # Now check to see what labels NLTK provides for the word
                    name_tag = word_tokenize(ent.text)
```

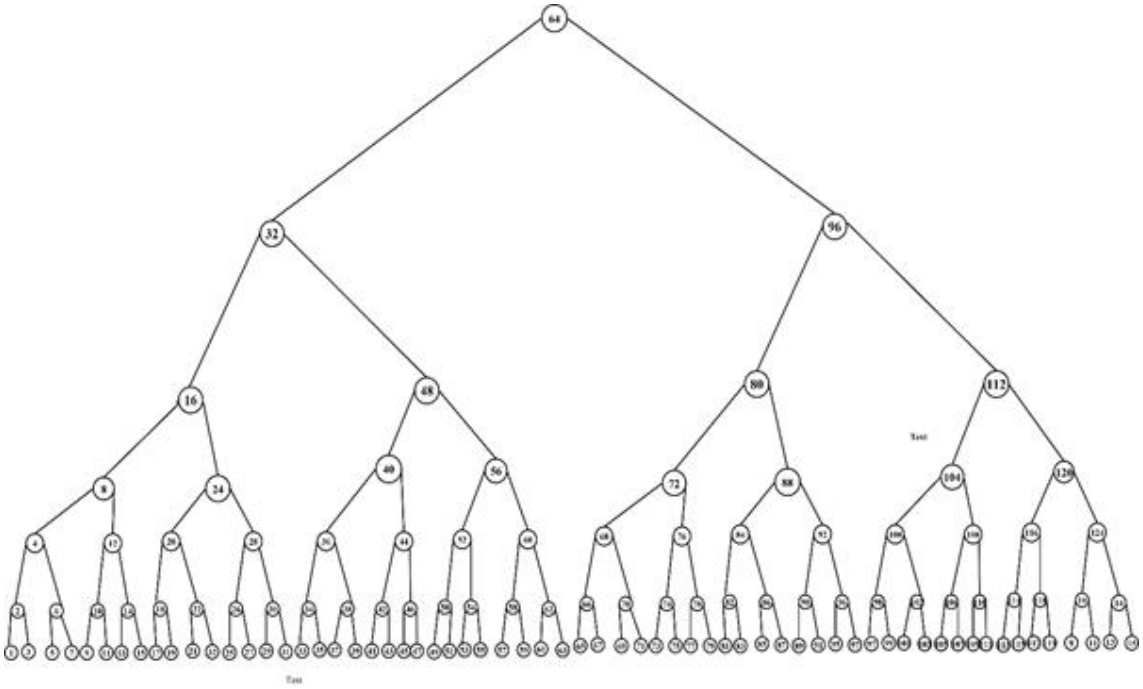
```
                # senna & hunpos
                #name_tag = pretrain.tag(name_tag)
                # hunpos needs to change the type from bytes to string
                #for j in range(len(name_tag)):
                    #name_tag[j] = list(name_tag[j])
                    #name_tag[j][1] = name_tag[j][1].decode('utf-8')
                #chunked = ne_chunk(name_tag)
                # default
                name_tag = pos_tag_sents([name_tag])
                chunked = ne_chunk(name_tag[0])
                for i in chunked:
                    if type(i) == Tree: # if ne_chunck thinks chunk is NER, creates a tree structure were leaves are the words in the chunk (and their POS labels) and the trunk is the single NER label for the chunk
                        if i.label() == 'PERSON':
                            for token, pos in i.leaves():
                                if pos == 'NNP':
                                    name_set.add(token)

                        else:
                            for token, pos in i.leaves():
                                spcy_upper_output = nlp(token.upper())
                                if spcy_upper_output.ents != ():
                                    name_set.add(token)

        # BEGIN STEP 8: whitelist check
        # sent_tag is the nltk POS tagging for each word at the sentence level.
        for i in range(len(sent_tag[0])):
            # word contains the i-th word and it's POS tag
            word = sent_tag[0][i]
            # print(word)
            # word_output is just the raw word itself
            word_output = word[0]

            if word_output not in string.punctuation:
                word_check = str(pattern_word.sub('', word_output))
                #if word_check.title() in ['Dr', 'Mr', 'Mrs', 'Ms']:
                    #print(word_check)
                    # remove the speical chars
                try:
                    # word[1] is the pos tag of the word

                    if (((word[1] == 'NN' or word[1] == 'NNP') or
                        ((word[1] == 'NNS' or word[1] == 'NNPS') and word_check.istitle()))):
                        if word_check.lower() not in whitelist_dict:
                            screened_words.append(word_output)
                         #   word_output = "**"+applyReflection(word_output)+"**"# "**PHI9**"
                            safe = False
                        else:
                            # For words that are in whitelist, check to make sure that we have not identified them as names
                            if ((word_output.istitle() or word_output.isupper()) and
                                pattern_name.findall(word_output) != [] and
                                re.search(r'\b([A-Z])\b', word_check) is None):
                                word_output, name_set, screened_words, safe = namecheck(word_output, name_set, screened_words, safe)

                    # check day/year according to the month name
                    elif word[1] == 'CD':
                        if i > 2:
                            context_before = sent_tag[0][i-3:i]
                        else:
                            context_before = sent_tag[0][0:i]
                        if i <= len(sent_tag[0]) - 4:
```

```
                    context_after = sent_tag[0][i+1:i+4]
                else:
                    context_after = sent_tag[0][i+1:]
                #print(word_output, context_before+context_after)
                for j in (context_before + context_after):
                    if pattern_mname.search(j[0]) is not None:
                        screened_words.append(word_output)
                        #print(word_output)
                        #word_output = "**PHI**"
                        safe = False
                        break
            else:
                word_output, name_set, screened_words, safe = namecheck(word_output, name_set, screened_words, safe)


            except:
                print(word_output, sys.exc_info())
            if word_output.lower()[0] == '\'s':
                if phi_reduced[-7:] != '**PHI**':
                    phi_reduced = phi_reduced + word_output
                #print(word_output)
            else:
                phi_reduced = phi_reduced + ' ' + word_output
        # Format output for later use by eval.py
        else:
            if (i > 0 and sent_tag[0][i-1][0][-1] in string.punctuation and
                sent_tag[0][i-1][0][-1] != '*'):
                phi_reduced = phi_reduced + word_output
            elif word_output == '.' and sent_tag[0][i-1][0] in ['Dr', 'Mr', 'Mrs', 'Ms']:
                phi_reduced = phi_reduced + word_output
            else:
                phi_reduced = phi_reduced + ' ' + word_output
    #print(phi_reduced)

    # Begin Step 8: check middle initial and month name
    if pattern_mname.findall(phi_reduced) != []:
        for item in pattern_mname.findall(phi_reduced):
            screened_words.append(item[0])
     #   phi_reduced = pattern_mname.sub('**PHI**', phi_reduced)

    if pattern_middle.findall(phi_reduced) != []:
        for item in pattern_middle.findall(phi_reduced):
        #    print(item[0])
            screened_words.append(item[0])
     #   phi_reduced = pattern_middle.sub('**PHI** **PHI** ', phi_reduced)
    # print(phi_reduced)


    if not safe:
        phi_containing_records = 1

     #  with open('indexing.csv', 'w', newline='') as writeFile:
    #       writer = csv.writer(writeFile)
        for x in screened_words:
            # n = String.count(word)
            matches = re.finditer(x, phi_reduced)
            matches_positions = [match.start() for match in matches]
            for index in matches_positions:
              #   start_index = phi_reduced.find(x)

                if (index > 0):
```

```
                #if ("/" in x or "-" in x):
                  # x = x.replace ("/", " ")
                   #x = x.replace ("-", " ")
                   phi_reduced = phi_reduced.replace(x, " **"+applyReflection(x, index)+"** ",1)
    #                writer.writerow ([x, " **"+applyReflection(x, index)+"** ", str(index),str(len(x))])
            else:
                   phi_reduced = phi_reduced.replace(x, " **"+applyReflection(x, index)+"** ",1)
    #                writer.writerow ([x, " **"+applyReflection(x, index)+"** ", str(index),str(len(x))])
    #    writeFile.close()
    #save phi_reduced file
    filename = '.'.join(tail.split('.')[:-1])+"_" + key_name + ".txt"
    filepath = os.path.join(foutpath, filename)
    with open(filepath, "w") as phi_reduced_note:
       print (filepath)
       phi_reduced_note.write(phi_reduced)


    filepath = os.path.join(foutpath,'filter_summary.txt')
    #print(filepath)
    screened_words = list(filter(lambda a: '**PHI' not in a, screened_words))
    #screened_words = list(filter(lambda a: a != '**PHI**', screened_words))

    with open(filepath, 'a') as fout:
       fout.write('.'.join(tail.split('.')[:-1])+' ' + str(len(screened_words)) +
          ' ' + ' '.join(screened_words)+'\n')
       # fout.write(' '.join(screened_words))

    print(total_records, f, "--- %s seconds ---" % (time.time() - start_time_single))
    # hunpos needs to close session
    #pretrain.close()
    return total_records, phi_containing_records

def main():
   # get input/output/filename

   ap = argparse.ArgumentParser()
   ap.add_argument("-i", "--input", default="input_test/",
            help="Path to the directory or the file that contains the PHI note, the default is ./input_test/.",
            type=lambda x: is_valid_file(ap, x))
   ap.add_argument("-r", "--recursive", action = 'store_true', default = False,
            help="whether to read files in the input folder recursively.")
   ap.add_argument("-o", "--output", default="output_test/",
            help="Path to the directory to save the PHI-reduced notes in, the default is ./output_test/.",
            type=lambda x: is_valid_file(ap, x))
   ap.add_argument("-w", "--whitelist",
            #default=os.path.join(os.path.dirname(__file__), 'whitelist.pkl'),
            default=resource_filename(__name__, 'whitelist.pkl'),
            help="Path to the whitelist, the default is phireducer/whitelist.pkl")
   ap.add_argument("-n", "--name", default="phi_reduced",
            help="The key word of the output file name, the default is *_phi_reduced.txt.")
   ap.add_argument("-p", "--process", default=1, type=int,
            help="The number of processes to run simultaneously, the default is 1.")
   args = ap.parse_args()
   finpath = args.input
   foutpath = args.output
   key_name = args.name
   whitelist_file = args.whitelist
   process_number = args.process
   if_dir = os.path.isdir(finpath)
   start_time_all = time.time()
   if if_dir:
      print('input folder:', finpath)
```

```
        print('recursive?:', args.recursive)
    else:
        print('input file:', finpath)
        head, tail = os.path.split(finpath)
        # f_name = re.findall(r'[\w\d]+', tail)[0]
    print('output folder:', foutpath)
    print('Using whitelist:', whitelist_file)
    try:
        with open(whitelist_file, "rb") as fin:
            whitelist = pickle.load(fin)
        print('length of whitelist: {}'.format(len(whitelist)))
        if if_dir:
            print('phi_reduced file\'s name would be:', "*_"+key_name+".txt")
        else:
            print('phi_reduced file\'s name would be:', '.'.join(tail.split('.')[:-1])+"_"+key_name+".txt")
        print('run in {} process(es)'.format(process_number))
    except FileNotFoundError:
        print("No whitelist is found. The script will stop.")
        os._exit(0)
    filepath = os.path.join(foutpath,'filter_summary.txt')
    with open(filepath, 'w') as fout:
        fout.write("")
    # start multiprocess
    pool = Pool(processes=process_number)

   results_list = []
  filter_time = time.time()
    # apply_async() allows a worker to begin a new task before other works have completed their current task
    if os.path.isdir(finpath):
        if args.recursive:
            results = [pool.apply_async(filter_task, (f,)+(whitelist, foutpath, key_name)) for f in glob.glob
(finpath+"/**/*.txt", recursive=True)]
        else:
            results = [pool.apply_async(filter_task, (f,)+(whitelist, foutpath, key_name)) for f in glob.glob
(finpath+"/*.txt")]
    else:
        results = [pool.apply_async(filter_task, (f,)+(whitelist, foutpath, key_name)) for f in glob.glob(  finpath)]
    try:
        results_list = [r.get() for r in results]
                total_records, phi_containing_records = zip(*results_list)
        total_records = sum(total_records)
        phi_containing_records = sum(phi_containing_records)
        print("total records:", total_records, "--- %s seconds ---" % (time.time() - start_time_all))
        print('filter_time', "--- %s seconds ---" % (time.time() - filter_time))
        print('total records processed: {}'.format(total_records))
        print('num records with phi: {}'.format(phi_containing_records))
    except ValueError:
        print("No txt file in the input folder.")
        pass
    pool.close()
    pool.join()
    # close multiprocess
if __name__ == "__main__":
    multiprocessing.freeze_support()  # must run for windows
    main()
```

Appendix IV: Binary Tree Data Structure



Appendix V: Cryptography

This section first explores early and developmental work in the area of cryptography. In particular, symmetric key ciphers and popular encryption standards such as the DES and AES are discussed. Standard references for classical cryptanalysis are also indicated.

Cryptographic algorithms can be divided to classic and modern cipher. Classic ciphers are character-oriented ciphers which can only be used to encrypt text. These can be divided into substitution ciphers and transposition ciphers. In a substitution cipher, each plaintext character is replaced by another character, either using a fixed replacement structure (monoalphabetic ciphers) or variable replacement structure (polyalphabetic ciphers). In transposition ciphers, plaintext characters are shifted depending on a given mapping key [83].These ciphers are highly susceptible to a cryptanalysis attack [206]. However, there have been some recent enhancements[207][208] to these ciphers to overcome these attacks and to increase security and maintain performance. For example, Marzan and Sison [209]enhanced the key security of the Playfair Cipher Algorithm using a combination of a

16 × 16 matrix, XOR, two's complement, and bit swapping. Aung and Hal [210] combined a Vigenère cipher with an Affine cipher to increase the level diffusion and confusion properties. Elmogy et al. [211] have proposed a new character-oriented cipher based on ASCII Code and the relationship between plaintext characters. The cipher uses simple operations like addition and subtraction and each character is encrypted differently based on the position of the previous character to avoid a frequency analysis attack. However, the cipher can be further improved by adding a random key generator. Similarly, Yadav et al. [212] propose a new symmetric algorithm to use ASCII conversion and the length of the plaintext to create a square matrix. The key is randomly generated based on the order of the square matrix.

Modern ciphers on the other hand are bit-oriented ciphers that can be used to encrypt any form of data. Simple examples of these ciphers are XOR ciphers, rotation ciphers and S-boxes. Simple modern ciphers have led to a new form of cipher called a product cipher or round cipher [213]. Product ciphers combine two or more transformations such as S-box, permutation and modular arithmetic[214]. The concept of product ciphers introduced by Shanoun[215] establish two main properties for the design of cryptographic algorithms: confusion and diffusion. Confusion obscures the relationship between the plaintext and ciphertext, whereas diffusion dissipates the statistical structure of plaintext over the bulk of ciphertext. These two properties can be achieved by producing a product cipher with multiple iterations. Each iteration works by combining different transformations to construct a complex encryption function. There are various implementations of these techniques, such as DES [216], 3DES[217], AES[218] and BLOWFISH [219].

Modern encryption algorithms can be divided into two main categories according to the encryption scheme: symmetric and asymmetric key encryption. Symmetric key encryption uses one secret key to encrypt and decrypt data, whereas asymmetric key encryption requires two different keys (public and private keys).

Symmetric encryption algorithms can also be subcategorized into two groups block and stream ciphers. Block ciphers encrypt a fixed size of n-bits of plaintext at once (e.g., 64 bits) while stream ciphers encrypt 1 bit or byte of plaintext at a time [220]. The block cipher algorithm is preferred to the stream cipher for faster computations [221]. While symmetric key algorithms have been considered a cryptographic solution for emerging big data applications, the cumbersome key management and distribution of this approach does not provide a suitable level of scalability. Therefore, more lightweight and practical alternatives need to be developed [14].

Advanced Encryption Standard

AES, also known as Rijndael, is a symmetric key block encryption algorithm that can encrypt data blocks of 128 bits using symmetric keys of 128, 192, or 256 bits. The cipher was designed by Joan Daemen and Vincent Rijmen [96], who submitted the proposal for evaluation by the U.S. National Institute of Standards and Technology (NIST). In 2010, NIST declared AES as the successor of the Triple Data Encryption System (3DES), which was broadly applied previously [222].

The AES cipher is based on the substitution-permutation network principle, in which the plaintext bytes are substituted and permuted at each round. Each round of transformation consists of four different operations: Add Round Key, Substitute Byte, Mix Columns, and Shift Rows. The number of rounds (Nr) applied to the plaintext depends on the key size, where there are 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. Each round requires a separate 128-bit round key that is generated from the master key using the AES key schedule. **Error! Reference source not found.** provides an overview of the structure of the AES algorithm.

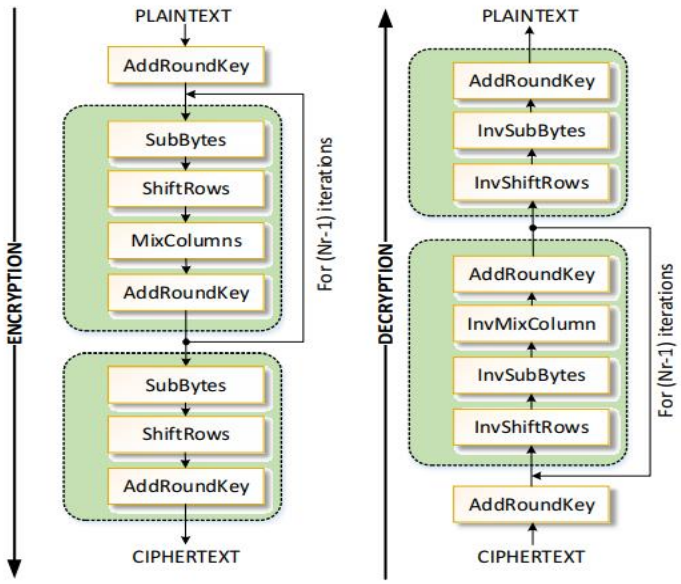


Figure 0-1 Encryption and Decryption Processes in AES [223]

Round Operations

As mentioned above, the AES encryption process consists of several operations that are repeated until the number of required rounds is reached. Before the round operations begin, the plaintext consisting of 128 bits is transformed into 16 bytes and stored in a 4×4 matrix, as shown in Figure 0-2. The resulting structure is known as a state array.

B_{00}	B_{04}	B_{08}	B_{12}
B_{01}	B_{05}	B_{09}	B_{13}
B_{02}	B_{06}	B_{10}	B_{14}
B_{03}	B_{07}	B_{11}	B_{15}

Figure 0-2: Data State Array

Add Round Key

In the Add Round Key operation, the state (or the plaintext) is added with the corresponding round key using exclusive OR (XOR), as illustrated in Figure 0-3. At

the start of the encryption process, and before the algorithm proceeds with the regular round operations, Add Round Key is performed once with the secret key.

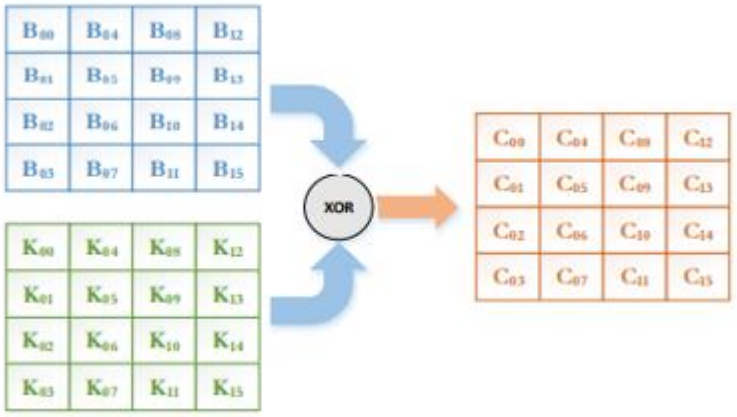


Figure 0-3: Add Round Operation [224]

Substitute Byte

In Substitute Byte, every byte in the state is mapped to a new byte based on the Substitution Box (S-Box). The S-Box is used to ensure Shannon's property of confusion by obscuring the relationship between the key and the ciphertext. The AES S-Box is often represented as a 16×16 hexadecimal matrix in which the rows represent the leftmost 4 bits and the columns represent the rightmost bits. The new mapped value is an intersection between the row and column. Figure 0-4 illustrates the process involved in Substitute Byte.

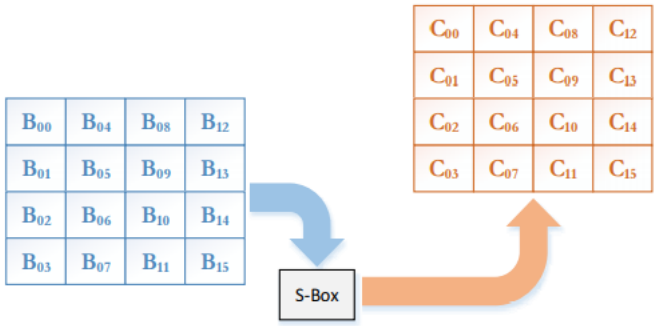


Figure 0-4: Substitute Byte Operation [224]

Shift Rows

This operation merely shifts bytes in the current state. The first row is left untouched while the second, third, and fourth rows are shifted with an offset of 1, 2, and 3 bytes, respectively, as illustrated in Figure 0-5.

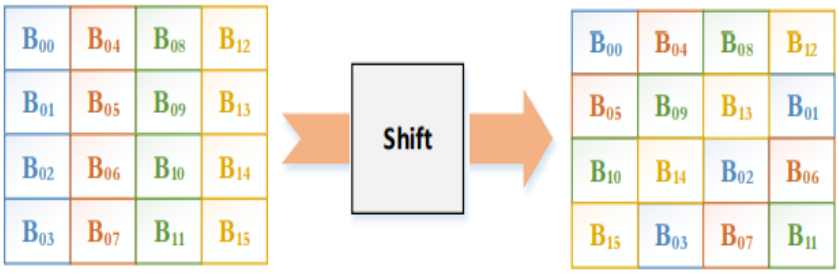


Figure 0-5: Shift Rows Operation [224]

Mix Columns

In Mix Columns, each column in the state is multiplied by a row of 5 4×4 matrices, as shown in Figure 0-6. The multiplication is performed over a Galois Field (GF), in which multiplying by 1 means doing nothing, multiplying by 2 means shifting to the left, and multiplying by 3 means shifting to the left and XORing with the operand. Taken together, the aforementioned operations – namely, Add Round Key, Substitute Byte, Mix Columns, and Shift Rows – provide the required diffusion for the AES encryption process.

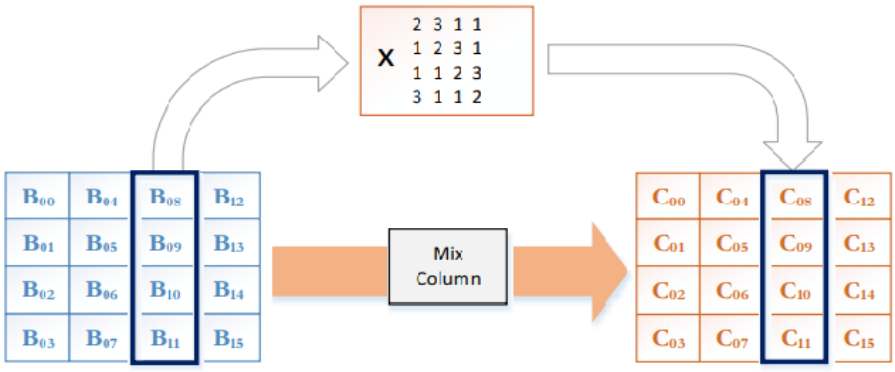


Figure 0-6: Mix Columns Operation [224]

Data Encryption Standard

DES is a symmetric key block encryption algorithm that can encrypt data blocks of 64 bits. It was created in the early 1970s by an IBM team and subsequently adopted by the NIST. It is based on a 56-bit secret key and 16 Feistel iterations surrounded by two permutation layers: initial permutation (IP) and final permutation (FP). The encryption process is illustrated in Figure 0-7 and can be summarised as follows:

1) The encryption process begins with the IP, which takes a 64-bit input and permutes it according to a predefined rule.
2) The IP generates two halves of the permuted block: 32-bit left (L) and 32-bit right (R).
3) Each half goes through 16 Feistel function iterations of the encryption process.
4) Finally, both the L and R bits are joined and the FP (i.e., the inverse of IP) is performed.

Figure 0-7: DES Encryption Function [225] shows the elements of the DES cipher at the encryption site. The Feistel function shown in Figure 0-8 is key-dependent and operates on 32 bits. It consists of four stages, which are described in the following subsections.

Expansion

The 32-bit input is expanded to a 48-bit input by redistributing and ordering the bits. The bits are selected from a pre-defined table. The rows in the table correspond to bits in the data, and the last bits of the data are taken from the input bits of the table.

Key Mixing

The bits generated in the first step are XORed with a round key that is constructed from 48 bits of a 56-bit input key. A different key is chosen for each round.

Substitution

The 48-bit result of the previous step is separated into 8 S-boxes and 6 bits words. Each of these 6-bit words is further substituted into 8 parallel 6×4 S-boxes. Each box is different but has the same structure, and the specifications are provided in a pre-defined table.

Permutation

A 32-bit word results from the previous step. These bits are reordered to a fixed permutation given in a predefined table. As before, the first row of the table refers to the first 4 bits of the output.

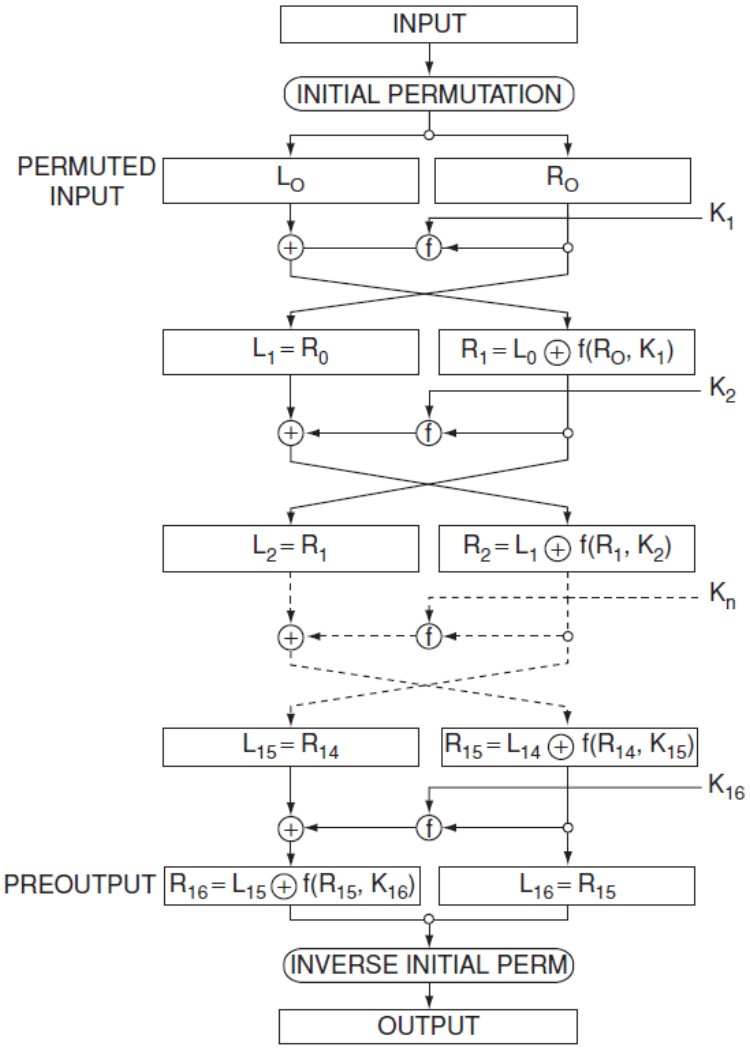


Figure 0-7: DES Encryption Function [225]

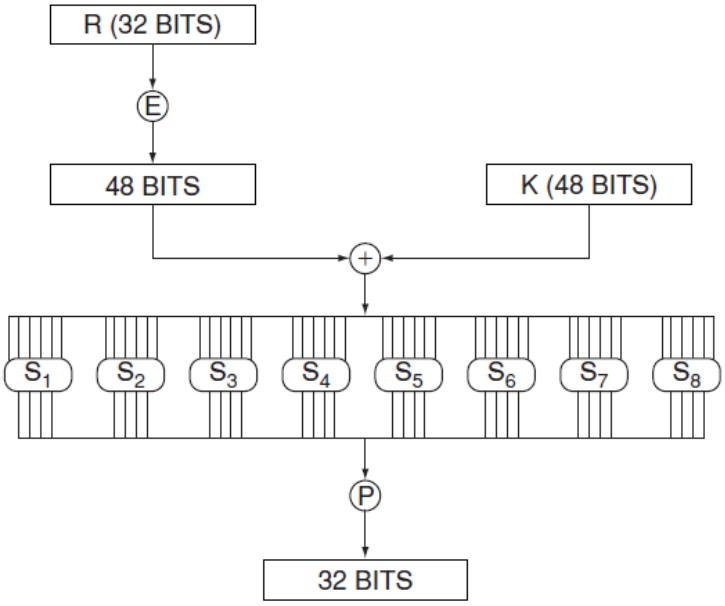


Figure 0-8: DES Function *f* [226]

Security against Attacks

A cryptographic attack is a means of breaking a cryptographic system's security by identifying a flaw in a code, cipher, or key management mechanism. This procedure is also known as cryptanalysis. A description of some known attacks is given in the following section.

Brute Force Attack

A brute force attack, also known as an exhaustive search, is an attack in which every potential key is tried in an attempt to decrypt the code. Theoretically, brute force attacks can work against almost every symmetric encryption algorithm, but in practice, these attacks are generally ineffective. This is because as the key size increases, brute force attacks become time-consuming. For example, if an adversary knows that the key size involved in a cryptosystem is 10 bits long, they can examine 2^{10} different keys, which is feasible. By contrast, if the key length is 128 bits, it is necessary to try 2^{128} potential keys, which is widely considered out of reach even for today's most advanced

technology. In the latter case, even a computer capable of executing billions of operations per second would require billions of years to examine every key [91].

Differential Cryptanalysis

Differential cryptanalysis was introduced in 1990 by Biham and Shamir [92]. It is a chosen-plaintext attack in which the adversary selects plaintexts and then performs encryption with the same key for each, where the key is unknown to the adversary. The plaintexts are in pairs in this context, and each plaintext has a specified difference relative to its paired plaintext.

Consider the two plaintexts $P, \check{P} = P \oplus X_p$ with their corresponding ciphertexts $Enc_k(P) = C,\ Enc_k(\check{P}) = \check{C}$. Let $X_{C=}\ C \oplus \check{C}$. If there exists a value X_C that occurs with a higher probability than would be if C and P were selected independently at random and not related through the encryption algorithm, then the cipher function can be differentiated from a random function. In this scenario, the differential relation is often used to retrieve part of the secret key.

$$\Pr\left[\ Enc_k(P)\ \oplus Enc_k\big(\ P\ \oplus\ X_p\ \big) = X_C\ \right] = \frac{1}{2} + \ \beta\ where\ \beta > \frac{1}{2} \qquad (0.3)$$

Meet-in-the-Middle Attack

The meet-in-the-middle attack was developed in 1977 by Diffie and Hellman, as explained in [93]. It is a type of known-plaintext attack in which the adversary knows some part of the plaintext and its corresponding ciphertext. The adversary tries to go from both ends (i.e., plaintext and ciphertext) to an intermediate state. Thus, the adversary encrypts the plaintext $Enc(P, k1)$ for a number of iterations, after which the ciphertext $Dec(C, k2)$ is decrypted for some iterations, to attain the same middle state of the cipher. This can be written as follows:

$$Enc(P, k1) = Dec(C, k2) \qquad (0.4)$$

Where C is the ciphertext corresponding to the plaintext P, both of which are known to the adversary.

The initial stage in the attack is to generate a table that contains all potential values for one side of eq. (2.4). In turn, the values are computed for the opposite side of eq. (2.4), after which they are compared to the values for the first side of the equation and saved in the table. The adversary looks for a pair of secret keys, $k1$ and $k2$, for which the value of $Enc(P, k1)$ in the table equals the computed value of $Dec(C, k2)$.