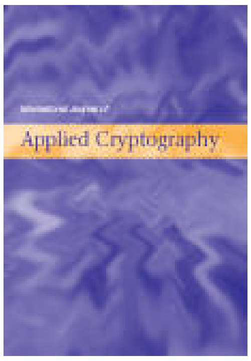


International Journal of Applied Cryptography

ISSN online: 1753-0571 - ISSN print: 1753-0563
https://www.inderscience.com/ijact

A novel keyless cryptosystem based on Latin square and cognitive artificial intelligence for blockchain and covert communications

Abdelrahman Desoky, Hany Ammar, Gamal Fahmy, Shaker El-Sappagh, Abdeltawab Hendawi, Sameh Hassanien Basha

DOI: 10.1504/IJACT.2024.10062638

Article History:

Received:	21 August 2023
Last revised:	25 November 2023
Accepted:	30 December 2023
Published online:	03 May 2024

A novel keyless cryptosystem based on Latin square and cognitive artificial intelligence for blockchain and covert communications

Abdelrahman Desoky*

Scired: Scientific Research and Development Corporation,
South Carolina, USA
and
Department of Computer Science,
Claflin University, USA
Email: desoky@desoky.com
*Corresponding author

Hany Ammar, Gamal Fahmy and
Shaker El-Sappagh

Faculty of Computer Science and Engineering,
Galala University, Egypt
Email: hany.ammar@gu.edu.eg
Email: gamal.fahmy@gu.edu.eg
Email: shaker.el-sappagh@gu.edu.eg

Abdeltawab Hendawi

Computer Science Department,
University of Rhode Island, USA
Email: hendawi@uri.edu

Sameh Hassanien Basha

Faculty of Science,
Cairo University,
Giza, Egypt
Email: samehbasha@cu.edu.eg
and
Faculty of Science,
Galala University,
Suez, Egypt
Email: samehbasha@gu.edu.eg

Abstract: The recent advances in cryptanalysis techniques and the leakage of information about the cryptosystem used are major threats to information systems. An adversary may succeed in decrypting ciphertexts, while users of a particular cryptosystem unknowingly continue using the compromised cryptosystem. Therefore, this paper presents a novel cryptosystem based on Latin square and cognitive AI/ML for blockchain and covert communications. This cryptosystem is capable of operating in quadruple modes keyless, symmetric, asymmetric, and hybrid encryption to cipher in cipher, and hence we call it CipherInCipher. Unlike all contemporary techniques including obscurity, CipherInCipher is a public-based approach that does not depend on the secrecy of any of its related components. It attains a high level of security that protects private information not only by having strong ciphertext but also by preventing an adversary from obtaining the actual ciphertext. The presented validation study demonstrates the robust CipherInCipher capabilities of achieving the cryptographic goal.

Keywords: cryptography; cryptosystem; cipher; ciphertext; security; secure communications; covert communications; blockchain.

Reference to this paper should be made as follows: Desoky, A., Ammar, H., Fahmy, G., El-Sappagh, S., Hendawi, A. and Basha, S.H. (2023) ‘A novel keyless cryptosystem based on Latin square and cognitive artificial intelligence for blockchain and covert communications’, *Int. J. Applied Cryptography*, Vol. 4, Nos. 3/4, pp.219–237.

Biographical notes: Abdelrahman Desoky is the CEO/Founder of Scired: Scientific Research and Development, and he is also an Associate Professor in the Department of Computer Science, at Claflin University. He has more than 20 years of experience in computer science/engineering, security, and related areas in both academia/industry. He was a visiting scholar at the University of California, Berkeley (2022). He received his Master of Science degree from George Washington University (2004) and Doctoral degree from the University of Maryland, Baltimore County (2009); both degrees in Computer Engineering. He is the author of a security book entitled *Noiseless Steganography: The Key to Covert Communications*.

Hany Ammar is a Professor of Computer Science and Engineering at Galala University, Egypt, and Professor Emeritus in the Lane Department of Computer Science and Electrical Engineering at West Virginia University, USA. His research interests are in software engineering and identification technology. He published more than 200 articles in prestigious international journals and conference proceedings. He served as the Lead PI and Co-PI in research projects funded by the US NASA, US NSF, US NIJ, and Qatar National Research Fund (QNRF). He has been teaching in the areas of software engineering and computer architecture since 1985.

Gamal Fahmy received his PhD in Electrical Engineering from Arizona State University, Tempe, USA in 2003. From 2003 to 2005, he was a Research Assistant Professor at West Virginia University, where he worked on several biometric identification and recognition projects; from 2006 to 2012, he was with the German University in Cairo and an annual summer senior researcher at the Institute of Image and Computer Vision at RWTH Aachen, Germany. He won the Egypt National State Award in Engineering Sciences 2012–2013. His research interests include image super-resolution, computer vision, biometric identification, and image forensics.

Shaker El-Sappagh received his PhD in Computer Science from the Information Systems Department, Faculty of Computers and Information, Mansura University, Mansura, Egypt in 2015. He worked as a Research Professor at UWB Wireless Communications Research Centre in the Department of Information and Communication Engineering at Inha University, South Korea for three years (2018–2020). He worked as a Research Professor at Universidade de Santiago de Compostela, Spain for one year (2021). Currently, he is an Associate Professor at Galala University, Egypt since 2021. He is a senior researcher at the College of Computing and Informatics, Sungkyunkwan University, South Korea since 2021.

Abdeltawab Hendawi is an Assistant Computer Science and Data Science Professor at the University of Rhode Island (URI). He is the Co-director of the AI-Lab at URI. He received his PhD in Computer Science from the University of Minnesota (UMN) in 2015. His research interests are centred on big data and AI, focusing on smart cities and smart health-related applications.

Sameh Hassanien Basha is a computational scientist with a PhD and Master's degrees from Cairo University, Egypt, specialises in machine learning, statistical methods, and optimisation techniques. His research centres on soft computing applications for handling imprecise data, utilising methods like the wavelet technique and developing neutrosophic rule-based systems (NRBS). In his PhD thesis, he introduced an evolutionary learning process to automate NRBS design, addressing their limitation in learning. His work extends to creating a hybrid system combining genetic algorithms and NRBS for genetic learning and knowledge base optimisation. His overarching research goal is advancing soft computing methods and machine learning algorithms.

This paper is a revised and expanded version of a paper entitled 'Latin square and artificial intelligence cryptography for blockchain and centralized systems' presented at the 9th International Conference on Advanced Intelligent Systems and Informatics (AISI2023), Springer, Port Said, Egypt, 20–22 September 2023.

1 Introduction and related work

Cryptography or cryptology is the science and art of concealing information, data, or both in an illegible format called ciphertext (Desoky et al., 2023; NIST1, 2022; NIST2, 2022). A ciphertext must be reversible back to its plaintext by only an authorised user via an operation called decryption (NIST3, 2022; Wade and Gill, 2022). Practically, the cryptographic goal is to avert an adversary from decrypting a ciphertext by generating a strong ciphertext. This can be done by utilising a strong algorithm that may include techniques and mathematical operations, etc. to generate a resilient ciphertext (Habib et al., 2022).

On the contrary, cryptanalysis is the science and art of contributing to breaching a cryptosystem to eventually decrypt its ciphertext (NIST4, 2022). This may be achieved, even if a cryptosystem is unknown, via studying whatever is available to start from such as ciphertext, some predicted or leaked information about a cryptosystem used (NIST5, 2022; Munir et al., 2022). Cryptanalysis may employ other techniques, tools, and applications such as math, statistics, arithmetic, reverse engineering, artificial intelligence, algorithms, hardware and software applications, etc. to perform its task (NIST5, 2022; Munir et al., 2022).

Yearly, the number of cybersecurity attacks is increasing worldwide (GAO, 2022; ALERT, 2022; DHS1, 2022; Security, 2022). Yet, with the current political tension and conflicts between countries worldwide, some cybersecurity attacks are directly done by governments or their supporters (DHS1, 2022; Security, 2022; DHS2, 2022). Once a government promotes attack(s), a virtually endless resource gets involved for a cybersecurity attack and cryptanalysis to decipher ciphertext to get valuable data and information.

Thus, tremendous dedication, obsession, and efforts motivate the recent advances in cryptanalysis techniques and the progressive leak of information about cryptosystems used are significant threats to cryptography (ALERT, 2022; DHS1, 2022; Security, 2022). Yet, an adversary may succeed in decrypting ciphertexts, while users of a particular cryptosystem unknowingly continue to use a compromised cryptosystem.

Contemporary cryptography approaches are mainly categorised into two types, symmetrical and asymmetrical key cryptography (Stallings, 2019; Koblitz, 1994). Symmetric key cryptography primarily refers to particular encryption techniques in which a legitimate communicating party shares the same secret key for encrypting and decrypting (Stallings, 2019; Koblitz, 1994). Asymmetric key cryptography such as a public key cryptosystem uses a pair of keys: a shared public key and an unshared private key (Stallings, 2019; Koblitz, 1994). Unlike symmetrical key encryption, a public key is shared and distributed publicly for anyone to use which resolves the key distribution problem (Stallings, 2019; Koblitz, 1994).

Some additional concerns of contemporary cryptography approaches may be summarised as follows. Most contemporary cryptography approaches have been used for a long time and this by itself is risky regardless of the name and type of any cryptosystem (Koblitz, 1994; COMPW, 2022). This is because every cryptosystem has its own life cycle time. Theoretically, the longer it is used the higher the probability of being breached because adversaries and researchers will have a good long time to contribute to breaching a cryptosystem (Koblitz, 1994). In addition, over time scientific research, applications, tools, software applications, computer systems, scientists, experts, and adversaries become more advanced and can contribute to breaching a secure cryptosystem. Yet, all users of contemporary cryptography approaches must use cryptographic keys regardless of the type of cryptosystem used. Moreover, the increasing number of phishing attacks and loss of key incidents became additional threats and vulnerabilities to contemporary cryptosystems too (DHS3, 2022).

The above issues led us to develop and present in this paper a novel cryptosystem based on Latin Square (LS) to Cipher In Cipher (CipherInCipher). Our proposed cryptosystem is for blockchain and covert communications capable of operating in quadruple modes keyless, symmetric, asymmetric, and hybrid encryption. Unlike all contemporary techniques including obscurity, CipherInCipher is a public-based approach that does not depend on the secrecy of any of its related components. It attains a high level of security that protects private information not only by having strong ciphertext but also by preventing an adversary from obtaining the actual ciphertext. The presented validation demonstrates the robust capabilities of achieving the cryptographic goal.

The remainder of this paper is organised as follows. Section 2 presents the CipherInCipher architecture. Section 3 demonstrates the implementation and the general validation of the proposed architecture. Section 4 discusses security validation and possible attacks. Finally, Section 5 concludes the paper.

2 CipherInCipher

To illustrate CipherInCipher, let us consider the following scenario. Bob and Alice work for a high-tech company, which involves a high level of security to protect sensitive information about new inventions and new products because of recent cyberattacks (ALERT, 2022). They considered the worst-case scenario where an adversary may be capable of decrypting all contemporary ciphertexts. Therefore, the CipherInCipher cryptosystem is selected to be utilised for this mission. This is because CipherInCipher is resiliently secure under the unfortunate assumption that an adversary is capable of breaking all contemporary cryptography algorithms. CipherInCipher attains a high level of security that protects private information not only by having strong ciphertext but also by making the actual ciphertext invisible to an adversary. When using CipherInCipher, the only full ciphertext that is normally visible to an adversary is a false ciphertext. This false ciphertext is either an unwanted cyphertext that conceals non-sensitive information or a fake ciphertext, but it is not the pure actual ciphertext. However, when using CipherInCipher a pure actual ciphertext is shredded into different sizes and it embeds it into either one relatively large false ciphertext or multiple relatively large false ciphertexts.

Note that the shredding and embedding procedures are done in such a sophisticated way based on a relatively large LS to make it more secure and to be extremely hard, if not impossible, for an adversary to reveal a plaintext (Laywine and Mullen, 1998; Dénes and Keedwell, 1991; Keedwell and Dénes, 2015).

Concisely, the CipherInCipher algorithm is achieved through three main phases as follows. First, CipherInCipher generates false ciphertext(s) by either legitimate ciphertext of unwanted plaintext or fake ciphertext. Second, it encrypts the intended message via symmetric, asymmetric, or hybrid encryption, as per user choice. Third, it embeds the encrypted message from phase 2 into the generated ciphertext from phase 1. Note that the embedding procedure is done according to a predetermined algorithm based on a relatively large LS with a minimum value of $N = 11$ to make it much more secure (Laywine and Mullen, 1998; Dénes and Keedwell, 1991).

It is worth noting that CipherInCipher gives legitimate users the option to operate in quadruple modes keyless, symmetric, asymmetric, and hybrid encryption by utilising a contemporary cryptography algorithm to secure communications, data, etc. as per users' choice in configuring its setup. One may say if we use a contemporary cryptography algorithm, why do we need to use CipherInCipher? The answer in short, as stated earlier CipherInCipher makes an actual ciphertext invisible to an adversary, as demonstrated and discussed in detail in the upcoming sections.

The presented cryptosystem architecture, implementation, and validation demonstrate the robustness capabilities of achieving the cryptographic goal, and the adequate room to achieve a high bitrate for concealing actual ciphertexts in false ciphertexts.

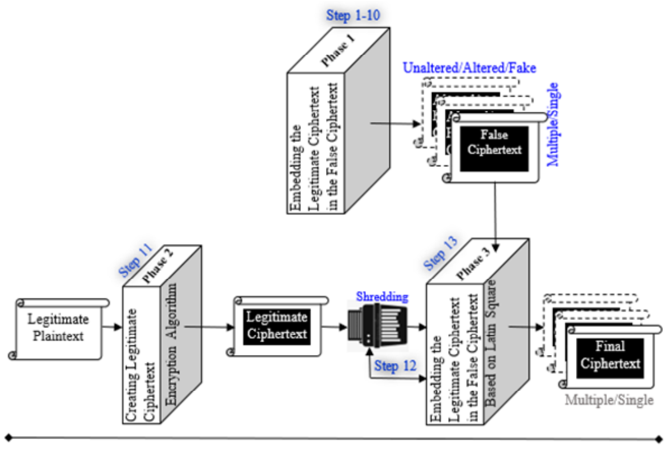
The following sections will discuss in more detail how each phase, procedure, and algorithm is done. Thus, the next section gives an overview of the CipherInCipher architecture.

2.1 *The CipherInCipher architecture*

The scenario of Bob and Alice in Section 2 demonstrates how the CipherInCipher paradigm can be utilised. This subsection will present a high level overview of the overall core architecture of the CipherInCipher system and how it achieves the security goal. We then present the detailed components of the architecture in the following subsections. Note that the implementation of the CipherInCipher system may differ from one implementation to another. Therefore, the following presents an overview of the main core options of the CipherInCipher algorithm and its architecture. Architecturally, the CipherInCipher system consists of three main phases, as shown in Figures 1, 2, and 3 along with Table 1 and the pseudocode in Algorithm 1. These three main phases are as follows:

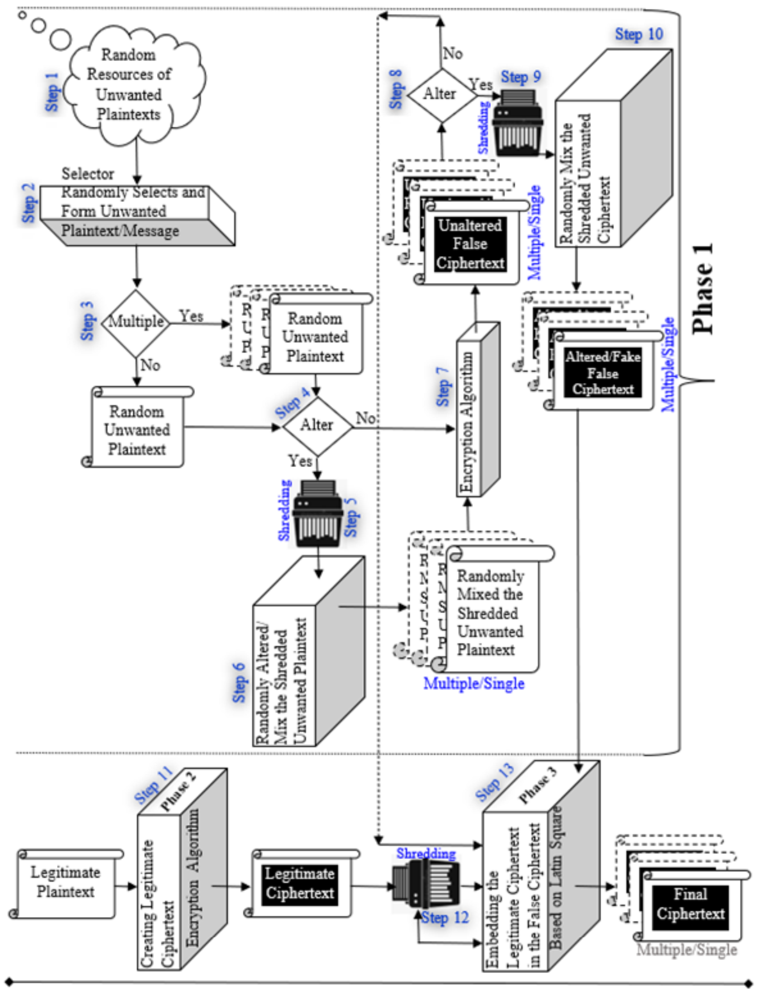
- *Phase 1:* CipherInCipher-based system generates false ciphertext by creating either a legitimate ciphertext of unwanted message/plaintext or fake ciphertext, as shown in Figures 1 and 2 and will be described in detail in Subsection 2.2 using Table 1 and the pseudocode in Algorithm 1. After ciphering the fake text, CipherInCipher uses a cognitive AI/ML model/system cryptoanalysis-based, internally, to ensure the robustness of the resulting ciphertext.
- *Phase 2:* CipherInCipher encrypts the intended message via symmetric, asymmetric, or hybrid encryption, as per user choice and its setup, as shown in Figures 1 and 2. This is done by using a contemporary cryptographic algorithm. This phase will be described further in Subsection 2.3. After ciphering of the sensitive text, CipherInCipher uses a cognitive AI/ML model/system cryptoanalysis-based, internally, to ensure the robustness of the resulting ciphertext.
- *Phase 3:* CipherInCipher embeds the encrypted message from phase 2 into the generated ciphertext from phase 1. Note that the embedding procedure is done according to a predetermined algorithm based on a relatively large LS with a minimum value of $N = 11$ to make it much more secure, as shown, mainly, in Figure 3 along with related Figures 1 and 2. This phase is described further in Subsection 2.4. After performing the embedding procedure of the legitimate ciphertext inside the false/fake ciphertext, CipherInCipher uses a cognitive AI/ML model/system cryptoanalysis-based, internally, to ensure the robustness of the resulting final ciphertext before it is used in covert communications.

Figure 1 Illustrates the architecture and the utilisation of CipherInCipher (see online version for colours)


Step 1-10
Phase 1
Embedding the Legitimate Ciphertext in the False Ciphertext
Unaltered/Altered/Fake
False Ciphertext
Multiple/Single
Step 11
Phase 2
Legitimate Plaintext
Creating Legitimate Ciphertext Encryption Algorithm
Legitimate Ciphertext
Shredding
Step 12
Step 13
Phase 3
Embedding the Legitimate Ciphertext in the False Ciphertext Based on Latin Square
Final Ciphertext
Multiple/Single

Notes: It shows the interaction of all phases 1, 2, and 3 with each other. Then, it shows the utilisation of the CipherInCipher system by the communication parties.

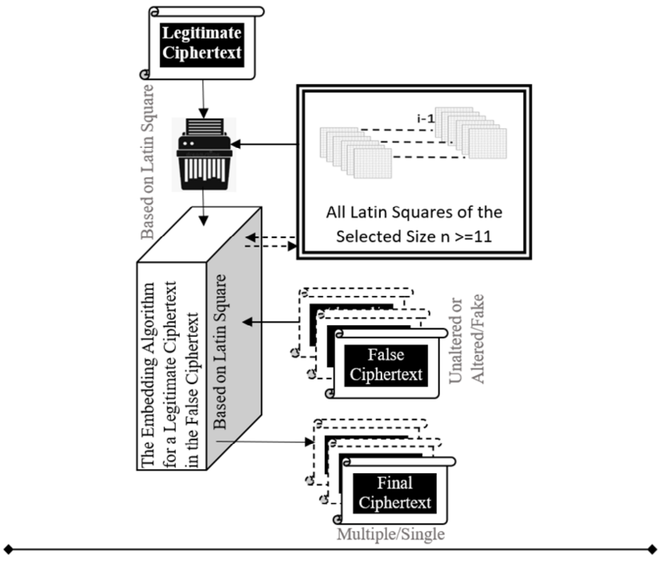
Figure 2 Illustrates the architecture and the utilisation of the CipherInCipher System with detailing of phase 1 (see online version for colours)



Step 1
Random Resources of Unwanted Plaintexts
Selector
Step 2
Randomly Selects and Form Unwanted Plaintext/Message
Step 3
Multiple
Yes
No
Random Unwanted Plaintext
Random Unwanted Plaintext
Step 4
Alter
No
Yes
Shredding
Step 5
Step 6
Randomly Altered/ Mix the Shredded Unwanted Plaintext
Randomly Mixed the Shredded Unwanted Plaintext
Multiple/Single
Step 7
Encryption Algorithm
Unaltered False Ciphertext
Multiple/Single
Step 8
Alter
No
Yes
Step 9
Shredding
Step 10
Randomly Mix the Shredded Unwanted Ciphertext
Altered/Fake False Ciphertext
Multiple/Single
Phase 1
Step 11
Phase 2
Legitimate Plaintext
Creating Legitimate Ciphertext
Encryption Algorithm
Legitimate Ciphertext
Shredding
Step 12
Step 13
Phase 3
Embedding the Legitimate Ciphertext in the False Ciphertext Based on Latin Square
Final Ciphertext
Multiple/Single

Notes: It shows the architecture and the interaction of various modules with each other in phase 1 to build a CipherInCipher-based system. Then, it shows the utilisation of the CipherInCipher system by the communication parties.

Figure 3 Illustrates the architectural part of the CipherInCipher system in phase 3 that utilises LS



Legitimate Ciphertext
Based on Latin Square
i-1
All Latin Squares of the Selected Size n >=11
The Embedding Algorithm for a Legitimate Ciphertext in the False Ciphertext Based on Latin Square
False Ciphertext
Unaltered or Altered/Fake
Final Ciphertext
Multiple/Single

2.2 Phase 1: generating false Ciphertext

The CipherInCipher-based system generates false ciphertext by creating either a legitimate ciphertext of an unwanted message (plaintext) or a fake ciphertext, as shown in Figures 1 and 2. The algorithm of phase 1, as shown in Figures 1 and 2, and Algorithm 1 along with Table 1, consists of several procedures (steps) that may reach up to ten procedures. Each procedure of Phase 1 will be executed in its independent step, as shown in both Table 1 and Algorithm 1 (the pseudocode).

Table 1 Shows the list of all steps of phase 1

Step # in phase 1	*What will do in each step (task detailed)*
Step 1	Provides Random resources of unwanted plaintexts. This is the initial input.
Step 2	Enables a selector that randomly selects unwanted plaintexts and forms a message(s) from step 1.
Step 3	Decides whether to utilise a multiple or single random unwanted plaintext(s). In this step, the CipherInCipher system will generate either multiple or single random unwanted plaintext(s) based on user setup and/or the implementation utilised and/or both together.
Step 4	Decides whether to utilise an unaltered or altered unwanted plaintext. If it is decided to utilise an unaltered of unwanted plaintext, then go to step 7, otherwise, go to the next step (step 5).
Step 5	If it is decided to utilise an altered of unwanted plaintext, then randomly shred unwanted plaintext(s).
Step 6	Randomly mix the shredded unwanted plaintext(s) to form a message or group of messages.
Step 7	Encrypts the output of step 4 or 6 using the selected contemporary cryptographic algorithm by the user or by the CipherInCipher system's setup, which includes several cryptosystems to be used for that.
Step 8	Decides whether to utilise an unaltered or altered Ciphertext. If it is decided to utilise unaltered of unwanted Ciphertext, then, go next (step 9), otherwise go to phase 3.
Step 9	If it is decided to utilise an altered ciphertext, then, the CipherInCipher system randomly shreds the ciphertext. Then go to the next step (step 10)
Step 10	Randomly mix the shredded ciphertext to form a fake ciphertext or group of fake ciphertexts that look(s) like legitimate ciphertext(s).
Step 11	Phase 2
Step 12	Shredding the output of phase 2 based on phase 3. In detail, step 12 shreds the output of phase 2 (step 11) based on phase 3, phase 3 setup, phase 3 implementation, and its LS.
Step 13	Phase 3

Phase 1 as stated before has up to ten steps. Steps 1 to 6 are responsible for preparing false plaintext to be encrypted to generate a false ciphertext, as follows. Step 1 is the initial input to the CipherInCipher system which provides random resources of false plaintexts (unwanted). Then, step 2 enables the selector to randomly select false plaintexts and forms an unwanted false message from step 1. Step 3 decides whether to utilise a multiple or single random false plaintext. In this step, the CipherInCipher system based on user setup and/or the implementation utilised and/or both together will generate either multiple or single random unwanted plaintexts.

Algorithm 1 The CipherInCipher algorithm pseudocode

```
BEGIN
Step 1: Provides random resources of unwanted plaintexts;
            // This step is an input.
Step 2: Enables a selector that randomly selects unwanted
          Plaintexts and forms a message(s);
          // From Step 1.
Step 3: IF multiple random unwanted plaintext(s) == True
            /* In this step, the CipherInCipher system will
            generate either multiple or single random
            unwanted plaintext(s) based on user setup and/or
            the implementation utilised and/or both together.
            */
          Then
            Generate the number needed of multiple random
            unwanted plaintext(s);
               // From step 2;
          Else
            Generate single random unwanted plaintext;
               // From Step 2;
Step 4: IF Altered unwanted plaintext == True
          Then
              Begin
Step 5:          Randomly shred unwanted plaintext(s);
                    // This step will generate altered
                    unwanted plaintext by shredding the
                    output of step 3.
Step 6:          Randomly mix the shredded unwanted
                 plaintext(s) to form a message or group of
                 messages;
                    /* The input of step 6 comes from step
                    5. However, based on step 3, it will
                    generate a single message or multiple
                    messages. */
              End
Step 7: Encrypt the output from the previous step (step 4 or 6);
                 /* The previous is a step 4 or 6. This is
                 done by using the selected contemporary
                 cryptographic algorithm by the user or by
                 the CipherInCipher system's setup. */
Step 8: If Altered Ciphertext == True
            Then
              Begin
Step 9:          Randomly shreds the Ciphertext;
Step 10:        Randomly mix the shredded ciphertext to
                form a fake ciphertext or group of fake
                ciphertexts;
/* This will be done to look like a legitimate ciphertext(s). */
            End
Step 11: Phase 2;
Step 12: Shredding the output of phase 2 based on phase 3;
/* Shredding the output of phase 2 (step 11) based on phase 3,
phase 3 setup, phase 3 implementation, and its Latin Square. */
Step 13: Phase 3;
END
```

Step 4 decides whether to utilise unaltered or altered false plaintext. The utilisation of altered false plaintext strengthens the final ciphertext that is generated by the CipherInCipher system. This is because for an adversary to decipher an encoded message and to claim successful deciphering, it must reach meaningful plaintext. So, when CipherInCipher randomly alters a false plaintext, it makes an adversary never reach meaningful plaintext. However, if CipherInCipher decides to utilise unaltered false plaintext, then CipherInCipher will escape steps 5 and 6 and will jump to step 7, otherwise, go to the next step (step 5). Reaching step 5 means it is decided to alter the false plaintext. Therefore, step 5 randomly shreds the false plaintext. Then, step 6 randomly mixes the shredded false plaintext to form a single false plaintext message or group of messages.

On the other hand, steps 7 to 10 are responsible for generating false ciphertext which will sooner be utilised in phase 3 for generating the final ciphertext that conceals the actual real ciphertext. Step 7 encrypts the output of step 4 or 6 using the selected contemporary cryptographic algorithm by the legitimate user or by the CipherInCipher system's setup, which includes several cryptosystems to be used for that. Then, step 8 decides whether to utilise unaltered or altered ciphertext. Therefore, in step 8 if it is decided to utilise an unaltered unwanted false ciphertext, then, go to next (step 9), otherwise, go to phase 3. Conversely, in step 8 if decided to exploit altered ciphertext, then, the CipherInCipher system performs step 9 which randomly shreds the ciphertext. Then, go to the next step (step 10). Step 10 randomly mixes the shredded ciphertext to form a single fake ciphertext or group of fake ciphertexts that look like a legitimate ciphertext. At this moment, phase 1 has ended, and the final output of phase 1 will be given to phase 3 as shown in Figures 1, 2, and Algorithm 1 along with Table 1.

2.3 *Phase 2: creating legitimate ciphertext*

Creating legitimate ciphertext is the procedure of encrypting an actual message. Given the availability of numerous encoding and cryptographic techniques in the contemporary literature that can suit CipherInCipher to be utilised (Koblitz, 1994), the balance of this paper will focus on the CipherInCipher system itself and will give examples of contemporary cryptography algorithms that can be utilised by CipherInCipher to demonstrate the applicability. Thus, in this paper, CipherInCipher will deal with the use of a contemporary cryptography algorithm, as if it is just a cryptographic box that receives messages as inputs and gives ciphertexts as outputs without too much detail about this box or the contemporary cryptography algorithm utilised. This is because it is not the actual contribution of this paper.

The intended users along with the CipherInCipher system requirement determine the appropriate contemporary cryptography algorithm to utilise for achieving the cryptographic goal. CipherInCipher can utilise symmetric, asymmetric, or hybrid encryption, as per user choice. This gives flexibility and freedom for selecting a contemporary cryptography algorithm that suits the requirements of both the CipherInCipher system and its users.

2.4 *Phase 3: embedding real ciphertext in cipher-cover*

Phase 3 embeds a legitimate ciphertext that is generated from phase 2 into a false ciphertext that is generated from phase 1, as shown via pseudocode in Algorithm 2. In other words, it embeds the encrypted message from phase 2 into the generated false ciphertext from phase 1. Note that a LS matrix with a size of N is an N × N matrix that has a core property that each symbol S occurs only once in each row and column (Laywine and Mullen, 1998; Dénes and Keedwell, 1991; Keedwell and Dénes, 2015). The embedding procedure is done according to a predetermined algorithm based on a relatively large LS with a minimum value of $N = 11$ to elevate its security. When LS uses a minimum value of $N = 11$ or greater than 11, will be an extremely large number of different unique LS as if it is a virtually infinite number of different unique LS (Laywine and Mullen, 1998; Dénes and Keedwell, 1991; Keedwell and Dénes, 2015; Van Lint and Wilson, 1992; Shao, 1992). Each one of the unique LSs will be utilised as a security key for the embedding procedure in phase 3. In this embedding procedure (phase 3), CipherInCipher will select only one LS, from a virtually endless number of different LS, to embed the legitimate ciphertext according to the selected LS. It is worth noting that the security key (the selected LS) is unknown even to legitimate users because the CipherInCipher system securely computes the LS/security key internally without sharing it with anyone. This makes CipherInCipher a keyless cryptographic algorithm, despite giving the users an option to be partially involved in the key procedure generation or selection if users choose to operate under such an option (setup). While the CipherInCipher algorithm is publicly known to all adversaries, it assures the security of its ciphertext from any illegitimate deciphering and prevents any negative leakage. For example, the CipherInCipher system ensures the secrecy of its security key(s) and prevents an adversary from concluding the LS/security key or any other information that may lead to breaking the CipherInCipher system via resilient mathematical computations, (e.g., LS, no-reuse keys, rotating procedures used, etc.), as will be shown in this section and the upcoming sections as well too. Note that the CipherInCipher system does not allow reusing an LS/security key once it is used. Furthermore, there are several ways to select a secure LS that plays the role of a security key in the CipherInCipher system, which will be discussed in this section and the next sections.

In this paper, due to space constraints, this section will discuss just a few concepts and examples in general of what and how the CipherInCipher system and its implementation may utilise, including the selection of secure LS, to achieve the security and cryptographic goal, as follows. For instance, how does CipherInCipher generate and utilise an LS/security key securely and internally? CipherInCipher system is a flexible system that may utilise the following concepts, some of them, or more than them. This confirms the fact that the CipherInCipher system may differ from one implementation to another.

- *Keyless:* CipherInCipher is a public algorithm which means, it is known to all adversaries. However, it assures the security of its ciphertext from any illegitimate deciphering and prevents any negative leakage. CipherInCipher system is a keyless system. Therefore, users neither hold keys nor use keys by themselves. However, if users want to use their keys, CipherInCipher can do that too, but it is not recommended due to the concern of losing a key, key leakage, stealing a key, key exchange, or similar issues.
- *LS security key:* CipherInCipher system internally and securely generates keys to be utilised without sharing them with anyone. So, users have no clue about any key used. For instance, one type of key that is internally, securely, and randomly generated and utilised is the LS with a minimum of $N = 11$ to elevate its security. It selects a specific LS to assign it to a group of users to use only one time. So, there is no reuse of any internal keys nor will be shared with users (e.g., LS, any other keys, etc.). CipherInCipher system will mark it to keep track of the used LS to avoid reusing keys. This includes other virtual keys such as the order or selection of mathematical operations utilised will be changed or rotated, etc. For example, the key for computing and generating the sequence of all LS is unknown to all users and there will be no reuse of such key as well. Mathematically and logically, the matrix of LS will be discussed in more detail according to its utilisation in the CipherInCipher system in the validation section (Section 3) (Laywine and Mullen, 1998; Dénes and Keedwell, 1991; Keedwell and Dénes, 2015; Van Lint and Wilson, 1992; Shao, 1992).
- *Other mathematical operations based on LS matrix:* concerning the mathematical operations, the CipherInCipher algorithm and its implementation can utilise a very sophisticated mathematical operation consisting of a group/combination of multiple mathematical operations that can be resiliently secure even when it is publicly known to an adversary, which is the case in CipherInCipher. Furthermore, a group/combination can be built using simple and cheap mathematical operations like addition, subtraction, multiplication, swap, shifting, and other operations, etc.

  However, the sequence order for a set of mathematical operations is based also on the LS matrix, which makes it in a dynamic order rather than just a static sequence. For example, when CipherInCipher utilises an LS-based matrix, CipherInCipher may select a row or column to apply a specific sequence order for a set of mathematical operations which will differ from one raw to another and one column to another too, according to properties of LS or/and partial LS utilised. Each cell of a raw or column in the LS matrix conveys a specific mathematical operation while that operation is unique in the entire raw and column according to LS property. Each time of communication will utilise a new row or column. Note that the CipherInCipher system will keep track of the sequence used for a particular group of users and it will differ each time used and from one group of users to another. The utilisation of the LS-based matrix will ensure the achievement of the security goal by changing the selected raw or column used each time via predetermined protocol internally randomly without sharing it with anyone. So, the CipherInCipher system will know/compute such sequence utilised but it will not be shared with anyone.
- *Fake users:* additionally, the CipherInCipher system generates many fake users in such a way as to look legitimate. This is in case of any internal attacks involved; adversaries will not be able to analyse it to conclude any useful information to break the CipherInCipher system.

The CipherInCipher system does not share any information with any user to ensure it is secure and unpredictable. Nonetheless, Section 3 will be more discussion and validation.

Algorithm 2 Pseudocode of the embedding procedure of phase 3 for the CipherInCipher algorithm

```
BEGIN
  If First_Time _Latin Square == True
  BEGIN
    Select Randomly Latin Square (LS) Where N =11;
    Keep Track of Selected LS;
  END
  Else
  BEGIN
      Select LS Based on the implemented equation
      Where N = 11;
/* For simplicity. Based on the implemented e.g., Selected LS is
number 129 then it can be 129+1, 129+3, 129+5, and so on.
This is just an easy example to make it clear, but it can be very
complex. */
      Keep track of Selected LS;
  END
```

```
    Keep track of the selected LS internally without sharing it
    with any users;
    Make the False ciphertext ready;
    // from Phase 1, as shown in detail in previous sections.
    Make the Real Ciphertext Ready;
        // from Phase 2, as shown in detail in previous sections.
    For J = 1; J <= Max # of Columns; J++;
        For i=1; i<=Max # of Columns; i++;
        BEGIN
            Read one LS Value at a time;
            IF (LS Value <= EOF False Ciphertext) && (Real
            Ciphertext NOT EOF);
    /* The false ciphertext is reasonable in size or to cover the
    real ciphertext or multiple false ciphertext will be utilised */
        BEGIN
        Go to the location that is equal to the LS value on the
        false ciphertext;
        Replace # of characters that matches the value of the
        selected LS with characters from the real ciphertext of
        phase 2;
    /* For example: if value = 3, go to position 3 then insert 3
    characters from the output of Phase 2, as shown in detail in
    previous sections. Remove number Replace */
        END
    END
END
```

2.5 *CipherInCipher decryption process*

The core idea of the presented paradigm shift in this paper, namely CipherInCipher as the name suggests, is concealing legitimate ciphertext in another fake/false ciphertext. Therefore, the reverse process is very simple and trivial to only legitimate users because the recipients will simply apply the reverse steps. The main challenge is revealing the legitimate ciphertext from the fake/false ciphertext. LS conveys the sequence that needs to be followed for collecting the legitimate ciphertext in the values of the actual LS used. Once a legitimate ciphertext is revealed, it will be directly decrypted by using the reverse of the actual CipherInCipher system used. It is worth mentioning that securely LS is utilised because it is relatively easy for legitimate users to create and keep tracking the index of which LS is used as invisible-key/keyless based on a predetermined protocol among users. Unlike any other mathematical model, it is extremely hard, if it is not impossible, for an adversary to hack the LS key used among virtual infinity (very large number) of LSs when $N = >11$ while securely enabling the concept of keyless, as explained mathematically and logically in Section 3.

3 Implementation validation

This section aims to demonstrate the implementation validation and feasibility of the CipherInCipher system and its distinct robustness capability of achieving the security and cryptographic goal, as follows.

3.1 *Implementation*

To avoid abstraction, this section demonstrates and discusses a possible implementation example and illustrates synthetic samples. The CipherInCipher system is flexible, and it can differ from one implementation to another and from one group of users to another. Thus, it can be built to make it harder for adversaries to attack the CipherInCipher system, as previously discussed. However, because of space constraints and for the sake of making it easier to understand and follow up with it, this section demonstrates easy examples as follows.

- *Sample of the output of phase 1:* CipherInCipher system generates false ciphertext by creating either a legitimate ciphertext of unwanted plaintext or fake ciphertext, to avoid repetition, as explained before and shown in Figure 4, 1, 2, Algorithm 1, Tables 1 and 2, along with online tools used in ENC1 (2022) and ENC2 (2022).

Figure 4 Illustrates the output of phase 1, which is the altered false cyphertext

```
105fb3JYNa3uCUW6X510gPxCK68JTJXd/hn2XSFEFfJ1v7r
LA+WzNSu9S38FjS81Zt01epcbrrtQcq8w6gxFPiUL6Ymmbr
y8+/rOlZOxNGNvM7A0EUUwmcx2dC4/noetl4A8KygBfZ0I
IH2aBiQlsIxGEp69weAoYnki5G1gKdI4Ay2ETjkNjXl/nuO1
aAs5yfrO7X+LgZxvdbwcUpbFDHU4mTkTdcSk9MSxkLkae
0wokhqXMPp2c17zrMcSmimKbZsILmTi6oNNj7VopSdE8k
eYlHY/AQDOmwxarcbvlOr+a6g1ykS6Cx5jtTPztW1YXiNe
LGvoS+w9UByBdc4IAjRMQMTBVWDLXqdfD7OoRt/Ff8L
and+ujFwei4TqS9S9P4lUqTPxqLzp048Isxp+zAToKw2sjQ/T
6Zab7cZILDAC
```

Note: This is produced after encrypting random unwanted plaintext.

Figure 5 Illustrates the output of phase 2, which is the legitimate cyphertext

```
hNYyDQwaB4dDqFhcfDW9O5tMJsyP53b9VcCx/OwStRY
```

Figure 6 Illustrates the output of phase 3, which is the final cyphertext of the CipherInCipher system that conceals the actual legitimate cyphertext

```
105fb3JhNYyDQwW6XaB4gPxCKdDqFhXd/hn2XSFcfDW9
O5tMA+JsNyu9S38FjSP53b9VcCcbrrtQcq8wStRYPiUL6Ym
mbry8+/rOlZOxNGNvM7A0EUUwmcx2dC4/noetl4A8KygB
fZ0IIH2aBiQlsIxGEp69weAoYnki5G1gKdI4Ay2ETjkNjXl/n
uO1aAs5yfrO7X+LgZxvdbwcUpbFDHU4mTkTdcSk9MSxk
Lkae0wokhqXMPp2c17zrMcSmimKbZsILmTi6oNNj7VopSd
E8keYlHY/AQDOmwxarcbvlOr+a6g1ykS6Cx5jtTPztW1YXi
NeLGvoS+w9UByBdc4IAjRMQMTBVWDLXqdfD7OoRt/F
f8Land+ujFwei4TqS9S9P4lUqTPxqLzp048Isxp+zAToKw2sj
Q/T6Zab7cZILDAC
```

- *Sample of the output of phase 2:* CipherInCipher encrypts the intended message via using a contemporary cryptographic such as symmetric, asymmetric, or hybrid encryption, as per user choice and its setup. This is as shown in Figures 5, 1, and 2, along with online tools used (ENC1, 2022; ENC2, 2022).

- *Sample of the output of phase 3:* CipherInCipher embeds the encrypted message from Phase 2 into the generated ciphertext from phase 1. Note that the embedding procedure is done according to a predetermined algorithm based on a relatively large LS with a minimum value of $N = 11$ to make it more secure, as shown in Figure 6 – 3, 1, 2, 4, and Table 2, along with online tools used (ENC1, 2022; ENC2, 2022; ENC3, 2022; ENC4, 2022). Note that LS is elaborated on in more detail in its immediate upcoming section (Section 3.2).

Table 2 Shows the selected LS used

7	3	5	9	2	1	8	4	10	11	6
1	8	10	3	7	6	2	9	4	5	11
9	5	7	11	4	3	10	6	1	2	8
3	10	1	5	9	8	4	11	6	7	2
8	4	6	10	3	2	9	5	11	1	7
11	7	9	2	6	5	1	8	3	4	10
10	6	8	1	5	4	11	7	2	3	9
5	1	3	7	11	10	6	2	8	9	4
6	2	4	8	1	11	7	3	9	10	5
4	11	2	6	10	9	5	1	7	8	3
2	9	11	4	8	7	3	10	5	6	1

3.2 *Mathematical analysis and proofs of utilising LS*

This subsection aims to show the mathematical complexity of how CipherInCipher takes advantage of the hard problem of computing a relatively big LS (e.g., n => 11).

Definition: first of all, LS matrix with a size of N is an N × N matrix that has a core property that each symbol S occurs only once in each row and column (Laywine and Mullen, 1998; Desoky, 2023). An $n \times n$ matrix is a LS only if it consists of n sets, (e.g., symbols, numbers, letters, a combination of all, etc.) arranged in such a way that no orthogonal (row or column) contains the same value more than one time (Van Lint and Wilson, 1992; Shao, 1992). Note that there are special cases of LS such as Latin Rectangle and partial LS that retain the same properties of LS (Laywine and Mullen, 1998; Dénes and Keedwell, 1991; Keedwell and Dénes, 2015; Van Lint and Wilson, 1992; Shao, 1992).

How many LSs are there? Mathematically, there is no easy computation method for the number L_n of $n \times n$ LSs with symbols 1, 2, …, n, as up to date. The most known upper and lower bounds assumed for large n are still vague. However, van Lint and Wilson stated a classical result that is never fully confirmed and is still fuzzy (Laywine and Mullen, 1998; Dénes and Keedwell, 1991; Keedwell and Dénes, 2015; Van Lint and Wilson, 1992; Shao, 1992). This classic result is that:

$$\prod_{k=1}^{n} (k!)^{\frac{n}{k}} \geq L_n \geq \frac{(n!)^{2n}}{n^{n^2}} \tag{1}$$

Shao and Wei reached a simple and explicit equation for calculating the number of LSs (Van Lint and Wilson, 1992; Shao, 1992). However, it is still a hard problem to compute because of the exponential increase in the number of terms. This equation for the number L_n of $n \times n$ LSs is (Van Lint and Wilson, 1992; Shao, 1992):

$$L_n = n! \sum_{A \in B_n} (-1)^{\sigma_0(A)} \binom{per\ A}{n} \tag{2}$$

where B_n is the set of all $n \times n$ {0, 1} matrices, $\sigma_0(A)$ is the number of zero entries in matrix A, and per(A) is the permanent of matrix A (Laywine and Mullen, 1998; Dénes and Keedwell, 1991; Disina et al., 2018; Keedwell and Dénes, 2015; Van Lint and Wilson, 1992; Shao, 1992).

It is worth noting that the online encyclopaedia of integer sequences (OEIS) was developed and maintained by Neil Sloane during his research at AT&T Labs. Then, it was transferred the intellectual property and hosting of the OEIS to the OEIS Foundation in 2009 (ENC3, 2022; ENC4, 2022). Table 3 shows all known exact values of all LSs of size n, as referenced via OEIS. It is observed that the numbers increase exceptionally rapidly. For instance, the number of all LSs registered a dramatic big jump between n and $n + 1$, especially starting when n is equal to 3, 4, 5, 6, and so on, as shown in Table 3. It is also observed that the number of all LSs holds a relatively very large number from $n = 6$ and so on. Once n reaches a value of 11, the number of all LSs becomes a virtual infinity number, as shown in Table 3. For each n, the number of LSs altogether (sequence A002860 in the OEIS), as shown in Table 3 (Laywine and Mullen, 1998; Dénes and Keedwell, 1991; Keedwell and Dénes, 2015; Van Lint and Wilson, 1992; Shao, 1992; ENC3, 2022; ENC4, 2022).

The usage of cell sequence in LS can be via a column-based, row-based, combination of both rows and columns together, and/or random-based, as predetermined. This will add more strength to the use of LS. Yet, if the usage level is just rows and columns, then, each LS will be used twice. Thus, the above large number will be multiplied by 2 which will create an extremely larger number than its original one. In addition, CipherInCipher can construct LS from any random size with a minimum $n = 11$ along with a random set of values that satisfy the properties of LS. Moreover, special cases of LS such as Latin Rectangle and partial LS can also be utilised by the CipherInCipher system which also will add more strength to it. Therefore, the prediction, calculation, brute force, and similar ways to attack are invisible to be used against CipherInCipher because the virtual infinity number of LS makes it extremely difficult for an adversary to deal with. For example, the task of searching or predicting a key among a virtually endless number of keys (e.g., LS used).

This makes LS with a relatively big n as at least $n = 11$ to be very promising in the field of cryptography in general and particularly in the presented cryptosystem in this paper, as shown in Table 3.

Table 3 Shows all LS of size N (sequences a002860 in the OEIS) increases by increasing the value of N

n	*All Latin squares of size n* *(Sequence A002860 in the OEIS)*
1	1
2	2
3	12
4	576
5	161, 280
6	812, 851, 200
7	61, 479, 419, 904, 000
8	108, 776, 032, 459, 082, 956, 800
9	5, 524, 751, 496, 156, 892, 842, 531, 225, 600
10	9, 982, 437, 658, 213, 039, 871, 725, 064, 756, 920, 320, 000
11	776, 966, 836, 171, 770, 144, 107, 444, 346, 734, 230, 682, 311, 065, 600, 000

3.3 Keyless

In this paper, the meaning of a keyless cryptosystem is that a user knows neither the actual key used nor its related information and there is no option given to users to hold or use a key. In this case, a user does not use a key by himself/herself to encrypt and/or decrypt. Therefore, it is called a keyless cryptosystem. Obviously, in every cryptosystem, either a key or something in it acts as a key regardless of whether it is key-based or keyless-based. Thus, in the CipherInCipher system, there is a use of keys but only internally covertly without sharing an actual key with any user or its related info.

In symmetric encryption, users share the use of a single key among communicating parties to encrypt and decrypt (Koblitz, 1994). On the other hand, in asymmetric encryption, each user has his unique pair of keys, a public key, and a private key to encrypt and decrypt messages (Koblitz, 1994). Conversely, the CipherInCipher system is a public keyless system, where users neither hold nor use keys by themselves. Therefore, CipherInCipher neither shares any key nor any related information to any key with any user. CipherInCipher can utilise centralised, decentralised, or hybrid centralised keyless management among communicating parties.

3.3.1 Centralised, decentralised, and hybrid

CipherInCipher system can utilise a centralised, decentralised, blockchain, or hybrid (combination of centralised and decentralised) keyless management system by taking advantage of LS proprieties, e.g., a large number of LS for $n = 11$, etc. to securely handle an LS-based keyless system, as discussed before (e.g., Subsection 2.4 in Section 2, Subsections 3.1 and 3.2 in Section 3), and as follows (Laywine and Mullen, 1998; Dénes and Keedwell, 1991; Keedwell and Dénes, 2015; Van Lint and Wilson, 1992; Shao, 1992; ENC3, 2022; ENC4, 2022).

3.4 Centralised system

A CipherInCipher centralised system and its all functionalities can be fully controlled by an individual, a group of people, or an entity like online social applications such as Facebook and Twitter (Liu et al., 2022).

CipherInCipher centralises the key management system (KMS) by conveying all its components and concepts such as hardware, software, lifecycle management, auditing, security, etc. into a central authority that consistently handles all issues of the keys (Stennikov et al., 2022). In this case, the KMS of CipherInCipher will be called a CKMS because it is centralised.

CipherInCipher utilises LS-based centric key generation management internally in a centralised system without sharing a key used or its information with any user, as explained before. It generates a new LS-based key for each time new communication because CipherInCipher does not allow the reuse of keys. Thus, it assigns the newly generated key to a specific group of users to use it only one time. The use of keys covertly and only internally without sharing them or their related information with users makes the CipherInCipher system a secure keyless system. This avoids problematic issues such as phishing attacks, losing keys, stealing keys, etc. In this case, the CipherInCipher centralised system gains higher performance, faster, and more dedication from the concepts of centralisation systems like the private blockchain.

3.5 Decentralised and blockchain system

A CipherInCipher decentralised system is an interconnected system in the form of networked computers rather than just a central authority (Kandi et al., 2022). Examples of decentralised architecture and systems are blockchain technologies such as Bitcoin and Ethereum (Stennikov et al., 2022; Kandi et al., 2022).

CipherInCipher decentralises the key management system (DKMS) and utilises and manages LS-based key generation to suit the decentralised system, where no central authority. Instead, CipherInCipher decentralises the KMS by leveraging the security, resiliency, availability, and immutability properties of distributed ledgers to provide vastly scalable DKMS (Kandi et al., 2022; ENC3, 2022; ENC4, 2022). One of the advantages of CipherInCipher is a decentralised LS-based key generation and management divides the risks into numerous nodes rather than just a single node. Thus, the CipherInCipher decentralised system is not vulnerable to a single node failure or attack. In this case, the CipherInCipher decentralised system gains the proven high security of the decentralisation systems concepts like a public blockchain.

3.6 *Hybrid*

In this paper, the CipherInCipher hybrid system is a combination of centralised and decentralised systems (WEB1, 2022a; Giron et al., 2023; Yu et al., 2015). It attempts to take advantage of both centralised and decentralised systems by combining similar structures of both systems into one structure system, called a hybrid system. It is similar to a centralised system from one viewpoint and a decentralised system from another (Islam et al., 2022; Wan et al., 2014). In some cases, the CipherInCipher hybrid system may look more like a decentralised system and vice versa depending on how much the implementation is leaning toward. In other words, more ingredients from either centralised or decentralised systems have a designated effect in its final look. Nonetheless, CipherInCipher attempts to employ the advantages of both centralised and decentralised systems and combine them into one system called hybrid (WEB1, 2022a; Islam et al., 2022).

Therefore, the CipherInCipher hybrid system gains the proven high security of decentralised systems like a public blockchain. Unlike the decentralised systems, it also gains higher performance, faster, and more dedication from the centralised system like the private blockchain. Technically, this is done by utilising a centric KMS that handles the key generation and managing it as a centralised system. Thus, it gains the advantages of a centralised system while gaining the advantages of a decentralised system.

3.7 *Honeynet-based fake nodes and users*

- *Definition and concept:* a honeynet is a decoy network or a subnetwork that contains one or more honeypots to trap, hunt, investigate, and learn about adversaries' activities to secure the actual network (WEB1, 2022b; Tan et al., 2021). It looks like a legitimate network or a subnetwork and contains multiple systems or subsystems but is hosted on one or only a few servers, each representing one environment such as a Windows-based honeypot, Mac-based honeypot machine, or Linux-based honeypot (WEB1, 2022b). Yet, it can also leak false information that fools adversaries (WEB1, 2022b; Tan et al., 2021).
- *CipherInCipher:* it utilises the concept of the honeynet-based system by employing fake nodes and users forming its honeypot(s) to fool adversaries and to keep all legitimate nodes and users safe and away from attacks as much as possible. Therefore, CipherInCipher creates a fake super node or number of fake super nodes and each one operates as a legitimate super node to fool an adversary. Additionally, it creates a fake regular node or number of regular nodes and each one operates as a legitimate regular node to fool an adversary. Fake nodes are involved in a process honeynet-based node. This will consume adversaries' resources distract adversaries and keep them away from the legitimate nodes. Fake nodes when acting as a honeynet allow studying adversaries' motives, techniques, etc. and any vulnerabilities that may exist to be fixed. Intentionally, it leaks false information in a convincing way that fools adversaries to keep the actual network safe.

3.8 *Cognitive artificial intelligence and machine learning based cryptanalysis*

In cryptanalysis, a cryptanalyst attempts to attack a real ciphertext to reveal a confidential message through the analysis and detection of hidden patterns in the ciphertext or predicting a secret key used. Our proposed cryptosystem is keyless and the actual ciphertext is invisible, as explained in detail before. As a result, this technique is secure by default. Recently, attackers used the recent cognitive artificial intelligence and machine learning (AI/ML) techniques to automatically discover confidential information in the ciphertext (Ahmadzadeh et al., 2022). To assure more robustness, CipherInCipher exploits cognitive AI/ML techniques to play the role of a cryptanalyst on the generated ciphertext to ensure the robustness of the generated cipher (Kim et al., 2023a).

There are potential roles that AI could play in the context of keyless ciphertext:

1 Designing algorithms: AI methods, such as evolutionary algorithms and neural networks, can be utilised to investigate and enhance encryption algorithms that do not rely on conventional keys. Researchers can leverage AI's pattern recognition capabilities to develop new encryption systems that use alternative techniques for secure communication.

2 Evaluating security: AI can be employed to assess the security of keyless encryption systems. Machine Learning models can analyse various keyless ciphertext strategies, identify weaknesses, and offer suggestions to improve security using AI-driven techniques.

3 Cryptanalysis: AI can assist in deciphering and analysing keyless encryption protocols. By training Machine Learning models on large datasets of encrypted data, patterns can be uncovered, potential flaws in encryption algorithms can be identified, and methods to decipher ciphertext without traditional keys may be discovered.

CipherInCipher inspects its generated ciphertext in the three phases to approve or disapprove the use of its outputs (e.g., false ciphertext, real ciphertext, and final ciphertext). In case of CipherInCipher system disapproves its ciphertext of any phases; CipherInCipher will redo the phase of the disapproved ciphertext to regenerate another ciphertext that will pass its inspection.

Many adversarial attacks could break the security of a ciphertext including (Luo and Chen, 2020; Baksi and Baksi, 2022; Tolba et al., 2022). To be robust against possible adversarial attacks, the CipherInCipher system attacks its own generated cyphertext to ensure it is secure before sending it to the legitimate recipient. As a result, CipherInCipher takes precautionary steps to prevent failures, and this is done intelligently and automatically using advanced cognitive AI/ML techniques such as residual connections and gated linear units. The problem is formulated as a Ciphertext classification task that learns the text using different machine learning, statistics, computational linguistics, information retrieval, and deep learning techniques. There are many applications of AI/ML in cryptanalysis (Andonov et al., 2020; Lagerhjelm, 2018; Kim et al., 2023b). Recently, different deep learning architectures have been proposed to optimise this task (Shin et al., 2020; Abd and Al-Janabi, 2019; Luthra and Pal, 2011). Different deep learning models have been used in cryptanalysis such as convolutional neural network (CNN) models have proved their capabilities in image analysis. CNN has been used in ciphertext analysis (Luthra and Pal, 2011; Alom et al., 2019), long short-term memory (LSTM), recurrent neural network (RNN), and gated recurrent unit (GRU) (Kumari et al., 2021; Rundo, 2020). While traditional cryptanalysis focuses on extracting the key to achieve success in deciphering ciphertexts, neural cryptanalysis aims to predict ciphertexts without prior knowledge of the key. In traditional cryptanalysis, a delicate human mathematical analysis is conducted to determine how the key value affects the statistics of plaintext-ciphertext pairs. The training data for neural cryptanalysis consists of pairs of plaintext and ciphertext, and the objective is to predict ciphertexts based on inputted plaintexts. This task is symmetric to predicting plaintexts from ciphertexts, as encryption and decryption are reversible processes. There have been only a few attempts to apply neural networks in the field of cryptography. In 1998, Clark (1998) introduced a neural network-based cryptography system, and in 2005, Kanter (2002) presented a method of using neural networks for secret key exchange via a public channel. Another proposal by Rao (2009) involved the use of a right-sigmoidal function as an activation function in a neural network for cryptanalysis of Feistel-type ciphers, but it did not yield any significant results. Measuring the strength of a cipher was explored by Xiao (2019), where the difficulty of replicating the cipher algorithm using a neural network was used as a metric. Additional metrics like cipher match rate, training data complexity, and training time complexity were introduced to numerically express cipher strength, enabling direct comparisons between different ciphers. The experimental approach involved fully connected neural networks with multiple parallel binary classifiers at the output layer.

However, the ciphertext sequence is decorrelated from the plaintext sequence. As a result, discriminative features between different categories are difficult to detect and require a deep understanding of the context (Rundo, 2020; Livieris et al., 2021). Other advanced deep learning techniques like bidirectional LSTM and transformers with attention mechanisms have a higher capacity to detect long dependencies to better understand the context (Livieris et al., 2021; Johnson and Zhang, 2016). We explore state-of-the-art machine learning and deep learning for detecting and highlighting the weakness of the generated cipher (Ahmadzadeh et al., 2022).

In addition, advanced ML and DL architectures such as ensemble DL models are proposed to enhance the state-of-the-art performance in measuring the quality of the cipher text and detecting the possible vulnerabilities. Different from plaintext learning, ML/DL must meet different criteria for efficient ciphertext interference. Language models like CNNs, long-short-term memory, and transformers are used to design high-performing and efficient models for ciphertext robustness evaluation. These models are extended using semantic and ontology knowledge to improve the interference power. Different hybridisation, optimisation (e.g., hyperparameters, architectures, and feature selection), and interpretability techniques are explored to improve the performance and explainability of the resulting model.

Note that ML/DL algorithms can be used in other steps in the proposed model to enhance its robustness and automation. For example, they can be used to determine the sensitive information in the plaintext input to phase 2. In addition, it can be used at the destination to detect possible attacks on the received messages. They can be used to generate random text that can be used as input to phase 1. ML/DL models can be optimised to generate ciphertext in the three steps. There is great potential for the application of AI/ML in the proposed architecture to prove its performance, robustness, and automation.

4 Security validation

This section aims to demonstrate the security validation, cryptoanalysis, and several cryptographic and cybersecurity attacks against the CipherInCipher system. The purpose is to show the distinct robustness capability of the CipherInCipher system in achieving the security and cryptographic goal, as follows.

4.1 Statistical signature

In this paper, a statistical signature and analysis refer to the pattern, fingerprint, profile, characters' frequency, etc. of ciphertext (Kim et al., 2023a; Siegenthaler, 1985; Li and Zhang, 2022). Numerically, a statistical signature is the number of times characters appear on average in a particular type of ciphertext (Kim et al., 2023a). Unlike the letters' frequency of a particular normal language, characters' frequency may contain more than just alphabetic letters and numerical values, like special symbols, etc. The frequency of characters and their plotted graph may vary from one type of ciphertext to another.

In CipherInCipher, the generated ciphertext is protected from such an attack because the actual ciphertext is embedded into a relatively large fake ciphertext. An actual ciphertext is a subset of the fake ciphertext. The size of an actual ciphertext is minimal compared to the size of its false ciphertext. Therefore, a statistical signature will mainly represent a false ciphertext used rather than its embedded actual ciphertext. Furthermore, the procedures in phases 1 and 2 of the CipherInCipher system kill any statistical signatures of any type of ciphertext used due to the use of several cryptosystems and the alteration procedures. Thus, it is infeasible for an adversary to accomplish a successful attack via statistical signatures against the CipherInCipher system.

4.2 *Known-plaintext analysis (KPA)*

In a KPA attack, an adversary attempts to obtain some known plaintext-ciphertext pairs to hack them to conclude a cryptographic key used (Li and Zhang, 2022; Nakano and Suzuki, 2022; Nakano et al., 2014). An adversary prevails if a KPA attack results in concluding the cryptographic key used. The strength of this attack may vary from one cryptographic approach to another and from an amount of available information to another. The availability of an adequate amount of information plays a major role in easing the KPA attack (Nakano et al., 2014).

Unlike all cryptosystems, in the CipherInCipher an actual ciphertext is not available to an adversary because CipherInCipher conceals an actual ciphertext in one or several false ciphertexts. Only, the false ciphertext may be available to an adversary. Additionally, the CipherInCipher system generates a new key for each time of communication and does not reuse the key more than once. Therefore, even if an adversary concludes a key used, is infeasible, still cannot be counted as a big victory. The reason is that a hacked-key cannot be reused to decrypt any future ciphertext due to the expiration of a key once it is used. This is because the applied policy of CipherInCipher is that a newly generated key only for new communication is to be used only one time and expires once used.

4.3 *Chosen-plaintext analysis (CPA)*

A CPA attack is based on the assumption that an adversary is capable of making a successful attempt to correspond ciphertexts to the selected arbitrary plaintexts or vice versa, and then, an adversary makes all practical and scientific efforts attempting to find the cryptographic key used (Anderson, 2020; Barrera et al., 2010; Rao and Cui, 2022). The practicality of the CPA attack is questionable and the probability of prevailing in exploiting a CPA attack is quite low according to its low rate of success (Anderson, 2020; Barrera et al., 2010; Rao and Cui, 2022). One of the goals of this attack is to gain more information about a particular cryptosystem to at least reduce its strength to make it vulnerable and insecure.

Unlike all obscurity-based technologies, CipherInCipher is a public-based approach that does not depend on the secrecy of any of its related components. CipherInCipher is resiliently secure under the assumption that an adversary has full information about CipherInCipher. Therefore, an adversary that applies a CPA attack and collects information about CipherInCipher as much as possible does not degrade its security. A CPA attack requires that an adversary be capable of corresponding ciphertexts to the selected arbitrary plaintexts or vice versa. This is not feasible at all when using CipherInCipher because the actual ciphertext is not available to an adversary. The only ciphertext that may be available to an adversary is the false ciphertext.

4.4 *Ciphertext-only analysis (COA)*

In cryptanalysis, a COA is based on the assumption that an adversary has only ciphertext due to his passive capability to eavesdrop and intercept communicating parties, etc. (Biryukov and Kushilevitz, 1998; Liao et al., 2021). The attacker only knows ciphertexts but not the corresponding plaintexts. However, an adversary with the capability to eavesdrop, intercept communication, etc. most likely has some advanced expert knowledge and tools that are not available to ordinary people. Some of the trivial information such as the language of plaintext that is concealed in the ciphertext and statistical signatures, etc. (Biryukov and Kushilevitz, 1998; Liao et al., 2021; Al-Shareeda and Manickam, 2022).

Unlike all cryptosystems, in the CipherInCipher an actual ciphertext is concealed in one or numerous false ciphertexts. In other words, the procedures of eavesdropping, intercepting, etc. are not capable of revealing the actual ciphertexts because as we explained before the actual ciphertexts are invisible since they are camouflaged in false ciphertexts. An adversary first has to find and collect all tiny pieces and bits of actual ciphertexts in a correct sequence and then, work on the actual ciphertext to decipher it. This is infeasible in our case since this is an extremely hard and very tedious operation if it is not impossible. In general, a COA attack is extremely hard and has a low rate of success. In addition, the applied policy of CipherInCipher key-expiration, once a key is used, does not allow an adversary to reuse a key if the key is successfully hacked.

4.5 *Man-in-the-middle (MITM)*

In network security, a MITM, as the linguistic meaning of its name, is an attack where an adversary is capable of getting between communicating parties impersonating a legitimate one and communicating via new messages or altering victims' communication (Fisher and Valenta, 2019; Fassl, 2018). It means an adversary must be able to encrypt, decrypt, and pass an authentication procedure as well

(Biryukov and Kushilevitz, 1998; Fisher and Valenta, 2019; Fassl, 2018).

In CipherInCipher, an MITM attack is inapplicable because it is not feasible that an adversary will encrypt, decrypt, and continue using a security key, as discussed before. First, the actual ciphertext is invisible to an adversary. Second, CipherInCipher is a keyless cryptosystem where the users do not use keys. This makes CipherInCipher immune from stealing key activities.

4.6 *Adaptive chosen-plaintext analysis (ACPA)*

An ACPA attack is like the CPA with the assumption that an adversary has access to an actual cryptosystem (Hu et al., 2014; Bard, 2006, 2007). This attack is very strong if all its requirements are satisfied. However, if an ACPA attack occurs successfully, it means, most likely there is an internal leakage and/or compromise such as an internal attack.

CipherInCipher achieves a high level of security that protects confidential information by concealing it in an invisible strong ciphertext which makes it infeasible for an adversary to even obtain an actual ciphertext. Yet, it utilises the concept of honeynet-based technology by employing fake nodes and users to form its honeypot(s) that fool adversaries and keep all legitimate nodes and users' safe and away from harm's way of cyberattacks as much as possible. Therefore, even with an internal leakage or compromise, our proposed system will still resist such circumstances.

4.7 *Internal attack*

One of the most severe attacks is the internal attack because an adversary, in this case, is a legitimate member of the organisation (Kalinin and Skvortsov, 2021). Thus, an internal attacker can have confidential information about the organisation and access to its hardware, software, and information, which can easily be used against the organisation to harm it than an external attack (Kalinin and Skvortsov, 2021).

Unlike all contemporary cryptography techniques including obscurity, CipherInCipher is a public approach that does not depend on the secrecy of any of its components and makes its actual ciphertext indiscernible to an adversary. This prevents an adversary from even obtaining the actual ciphertext which is essential for any attack to prevail.

Furthermore, CipherInCipher utilises the concept of honeynet-based technology, as discussed in Subsection 3.7 of Section 3, by employing fake nodes and users to form its honeypot(s) that fools adversaries and keep all legitimate nodes and users safe and away from harm's way of cyberattacks as much as possible (WEB1, 2022b; Tan et al., 2021). Therefore, even if an internal leakage or compromise occurs, it will still resist under such circumstances. This contributes to the security of CipherInCipher if an internal attack occurs because it can fool an adversary. In case of big security breaches such as severe leakage or even sending an entire copy of the actual CipherInCipher system to an adversary, it will still resist. This is because the actual CipherInCipher system contains false information and fake nodes and users that look legitimate to adversaries due to the utilisation of honeynet-based technology to fool an adversary as planned, as discussed before. Table 4 concludes a summary and comparison of all presented mentioned attacks.

Table 4 Summary and comparison of all presented attacks

Cryptosystem type →	*Symmetric*	*Asymmetric*	*CipherInCipher*
Attack ↓ / Output →	Actual ciphertext	Actual ciphertext	Overall false/fake ciphertext
Statistical signature	Feasible	Feasible	Infeasible
Known-plaintext analysis (KPA)	Feasible	Feasible	Infeasible
Chosen-plaintext analysis (CPA)	Feasible	Feasible	Infeasible
Ciphertext-only analysis (COA)	Feasible	Feasible	Infeasible
Man-in-the-middle (MITM)	Feasible	Feasible	Infeasible
Adaptive chosen-plaintext analysis (ACPA)	Feasible	Feasible	Infeasible
Internal attack	Feasible	Feasible	Infeasible
Brute force attack	Feasible	Feasible	Infeasible

4.8 *CipherInCipher vs. steganography*

Steganography is the science and art of camouflaging the presence of covert communications (Desoky, 2012). Fundamentally, the steganographic goal is not to hinder an adversary from decoding a hidden message, but to prevent an adversary from suspecting the existence of covert communications (Desoky, 2012). When using any steganographic technique if suspicion is raised, the goal of steganography is defeated regardless of whether or not a plaintext is revealed (Desoky, 2012).

Contemporary steganography approaches are often classified based on the steganographic cover type into image, audio, graph (Munir et al., 2022), or text. Textual steganography can be linguistic and non-linguistic. When linguistics is employed for hiding data and generating the steganographic cover, an approach is usually categorised as linguistic steganography to distinguish it from non-linguistic techniques, e.g., image, audio, graph, and text non-linguistic based. A text non-linguistic-based does not follow any structure and grammar rules of human language like URL-based Steganography. Therefore, the steganographic cover must look innocent to an adversary.

Conversely, cryptography is the science and art of encoding confidential data and information in an unreadable format called ciphertext (Stallings, 2019; Koblitz, 1994). A ciphertext should not be decryptable to its plaintext by an unauthorised user. Unlike a steganographic cover, a ciphertext advertises the fact that a valuable set of data or information is concealed. Therefore, a secure ciphertext depends on the strength of its cryptosystem that prevents an adversary from decrypting its ciphertext.

Fundamentally, the steganographic goal is not to hinder the adversary from decoding a hidden message, but to prevent an adversary from suspecting the existence of covert communications (Desoky, 2012). The set of challenges in the steganography area is not in the CipherInCipher approach where the goal is also different. CipherInCipher like any other cryptographic approach produces ciphertext that advertises the concealment of secrets.

It is worth noting that an embedding procedure of a message in a steganographic cover is a big challenge because it must look innocent to an adversary to avoid raising suspicion. However, in CipherInCipher the embedding procedure of ciphertext in another ciphertext, as explained before, is not a major concern like in steganography. This is because steganography avoids any illegible format that may raise suspicion, while any cyphertext is already an illegible format by its nature. Therefore, hiding ciphertext in another ciphertext is relatively easier than hiding it in a steganographic cover (Munir et al., 2022). Furthermore, the low bitrate in steganography is a major concern because increasing steganography bitrate most likely raises suspicion which will defeat the purpose of a steganographic approach (Desoky, 2012). Conversely, a bitrate in the CipherInCipher can vary from one implementation to another. However, the CipherInCipher bitrate most likely be higher than the steganographic bitrate. This is due to the severe sensitivity of steganographic cover for increasing a bitrate. On the other hand, CipherInCipher is relatively insensitive to increasing bitrate issues compared to steganography. It is due to the nature of CipherInCipher which conceals messages in an illegible format like ciphertext rather than a legible format like normal text. Thus, CipherInCipher along with all other validations and reasons that are stated before in this paper is a very promising approach for securing information and data whether in the static or the transmission stage.

5 Conclusions

In this paper, we presented the architecture and validation of a novel paradigm shift cryptosystem, namely CipherInCipher, based on a relatively large LS for blockchain and covert communications demonstrating the robust capabilities of achieving the cryptographic and security goal. Unlike all contemporary techniques including obscurity, CipherInCipher is a public-based approach that does not depend on the secrecy of any of its related components. It attains a high level of security that protects private information, data, or both not only by generating strong ciphertext but also by preventing an adversary from even obtaining the actual ciphertext by concealing it in a false ciphertext. CipherInCipher gives legitimate users the option to operate in triple modes symmetric, asymmetric, and hybrid encryption for the actual ciphertext. In addition, the CipherInCipher system is keyless for the main ciphertext that conceals an actual ciphertext. We presented the architecture, implementation, and validation of the approach to illustrate this promising approach for securing information and data whether in the static or the transmission stage.

The future work of this research should focus on applying the proposed approach to a large number of different domains such as the medical and financial application domains. These applications would use different sets of data types such as graphs, images, and audio files including big data sets and different application systems.

Acknowledgements

We would like to express our thanks to the well-known cryptographer Dr. Taher Elgamal for the great generosity of his support, effort, and time in discussing and advising in building the CipherInCipher and CAIMLSSI system.

This work was supported and funded fully by The Scired: Scientific Research and Development (Scired Corporation at https://scired.com) under TR/Project #0115202201 and TR/Project #0601201801.

Preliminary and shorter versions of this work appeared in the 9th International Conference on Advanced Intelligent Systems and Informatics (AISI2023), Springer, Desoky et al. Yet, this work is a patent pending/filed US PPAN 63/514,796 (Desoky, 2023).

Research data policy and data availability statements

All data, information, etc. are available in the public domain for anyone to use as referenced.

References

Abd, A.J. and Al-Janabi, S. (2019) 'Classification and identification of classical cipher type using artificial neural networks', *Journal of Engineering and Applied Sciences*, Vol. 14, No. 11, pp.3549–3556.

Ahmadzadeh, E., Kim, H., Jeong, O., Kim, N. and Moon, I. (2022) 'A deep bidirectional LSTM-GRU network model for automated ciphertext classification', *IEEE Access*, Vol. 10, pp.3228–3237, DOI: 10.1109/ACCESS.2022.3140342.

ALERT (2022) *Alert (AA22-040A) 2021 Trends Show Increased Globalized Threat of Ransomware*, Original release date: February 9, 2022, Last revised: February 10, 2022, [online] https://www.cisa.gov/uscert/ncas/alerts/aa22-040a (accessed 23 October 2022).

Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S. and Asari, V.K. (2019) 'A state-of-the-art survey on deep learning theory and architectures', *Electronics*, Vol. 8, No. 3, p.292.

Al-Shareeda, M.A. and Manickam, S. (2022) 'Man-in-the-middle attacks in mobile ad hoc networks (MANETs): analysis and evaluation', *Symmetry*, Vol. 14, No. 8, p.1543.

Anderson, R. (2020) *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley & Sons, Indianapolis, Indiana, USA.

Andonov, S., Dobreva, J., Lumburovska, L., Pavlov, S., Dimitrova, V. and Popovska Mitrovikj, A. (2020) 'Application of machine learning in DES cryptanalysis', *Proceedings of the 12th ICT Innovations Conference*, Skopje, North Macedonia, September.

Baksi, A. and Baksi, A. (2022) 'Machine learning-assisted differential distinguishers for lightweight ciphers', *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*, *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, IEEE Xplore, Singapore, November 2021, pp.141–162, DOI: 10.1109/VLSI-SoC53125.2021.9606988.

Bard, G.V. (2006) *A Challenging but Feasible Blockwise-Adaptive Chosen-Plaintext Attack on SSL*, Cryptology ePrint Archive.

Bard, G.V. (2007) 'Blockwise-adaptive chosen-plaintext attack and online modes of encryption', in *Cryptography and Coding: 11th IMA International Conference*, Springer Berlin Heidelberg, Cirencester, UK, 18–20 December, Proceedings 11, pp.129–151.

Barrera, J.F., Vargas, C., Tebaldi, M. and Torroba, R. (2010) 'Chosen-plaintext attack on a joint transform correlator encrypting system', *Optics Communications*, Vol. 283, No. 20, pp.3917–3921.

Biryukov, A. and Kushilevitz, E. (1998) 'From differential cryptanalysis to ciphertext-only attacks', in *Advances in Cryptology – CRYPTO'98: 18th Annual International Cryptology Conference*, Springer Berlin Heidelberg, Santa Barbara, California, USA 23–27 August, Proceedings 18, pp.72–88.

Clark, M.B. (1998) 'A neural-network based cryptographic system', in *Proceedings of the 9th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 1998)*, pp.91–94.

COMPW (2022) *The Clock is Ticking for Encryption: The Tidy World of Cryptography may be Upended by the Arrival of Quantum Computers* [online] https://www.computerworld.com/article/2550008/the-clock-is-ticking-for-encryption.html (accessed 3 August 2022).

Dénes, J. and Keedwell, A.D. (1991) *Latin Squares: New Developments in the Theory and Applications*, Vol. 46, Elsevier, The Netherlands.

Desoky et al. (2023) 'Latin square and artificial intelligence cryptography for blockchain and centralized systems', *9th International Conference on Advanced Intelligent Systems and Informatics (AISI2023)*, Springer, (Preliminary and short version).

Desoky, A. (2012) *Noiseless Steganography: The Key to Covert Communications*, CRC Press, USA.

Desoky, A. (2023) *Cipher in Cipher: A Novel Keyless Cryptosystem Based on Latin Square and Artificial Intelligence for Blockchain and Covert Communication*, US Patent Pending/Filed: US PPAN 63/514,796, July.

Desoky, A. and Ammar, H. (2022) *Keyless Cryptosystem Based on Latin Square for Blockchain and Covert Communications*, Scired Technical Report TR/Project # 0115202201, Scired: Scientific Research and Development (Scired Corporation) [online] https://scired.com (accessed 7 January 2023).

Desoky, A., Ammar, H., Fahmy, G., El-Sappagh, S., Basha, S.H. and Hendawi, A. (2023) 'Securing confidential information in big data based on cognitive artificial intelligence and machine learning', in *2023 International Conference on Artificial Intelligence Science and Applications in Industry and Society (CAISAIS)*, IEEE, September, pp.1–6.

DHS1 (2022) *The Department of Homeland Security: Cyber Incident Reporting: A Unified Message for Reporting to the Federal Government* [online] https://www.dhs.gov/sites/default/files/publications/Cyber%20Incident%20Reporting%20United%20Message.pdf (accessed 23 October 2022).

DHS2 (2022) *The Department of Homeland Security: Cybersecurity & Infrastructure Security Agency, Worldwide Cyber Incident Reporting* [online] https://www.cisa.gov/shields-up (accessed 23 October 2022).

DHS3 (2022) *The Department of Homeland Security: Cybersecurity & Infrastructure Security Agency, Cybersecurity Attacks, Incident Reporting* [online] https://www.cisa.gov/stopransomware/general-information (accessed 23 October 2022).

Disina, A.H., Jamel, S., Pindar, Z.A. and Deris, M.M. (2018) 'Statistical analysis, ciphertext only attack, improvement of generic quasigroup string transformation and dynamic string transformation', *Advanced Science Letters*, Vol. 24, No. 6, pp.4459–4463.

ENC1 (2022) *An Online Encryption* [online] https://www.online-toolz.com (accessed 31 August 2022).

ENC2 (2022) *An Online Encryption* [online] https://www.dcode.fr (accessed 31 August 2022).

ENC3 (2022) *An On-Line Encyclopedia of Integer Sequences (OEIS)* [online] https://oeis.org/wiki/Welcome (accessed 31 August 2022).

ENC4 (2022) *An On-Line Encyclopedia of Integer Sequences (OEIS)* [online] https://oeis.org/A002860 (accessed 31 August 2022).

Fassl, M. (2018) *Usable Authentication Ceremonies in Secure Instant Messaging*, Doctoral dissertation, Wien.

Fisher, G. and Valenta, L. (2019) *Monsters in the Middleboxes: Introducing Two New Tools for Detecting HTTPS Interception*, April, Springer, Cham, ISBN: 978-3030935733.

GAO (2022) *Government Accountability Office (GAO)*, *Report Incidents* [online] https://www.gao.gov/cybersecurity (accessed 23 October 2022).

Giron, A.A., Custódio, R. and Rodríguez-Henríquez, F. (2023) 'Post-quantum hybrid key exchange: a systematic mapping study', *Journal of Cryptographic Engineering*, Vol. 13, No. 1, pp.71–88.

Habib, H.B., Hussein, W.A. and Abdul-Rahman, A.K. (2022) 'A hybrid cryptosystem based on Latin square and the modified BB84 quantum key distribution', *Tikrit Journal of Pure Science*, Vol. 27, No. 4, pp.100–103.

Hu, W., Wu, L., Wang, A., Xie, X., Zhu, Z. and Luo, S. (2014) 'Adaptive chosen-plaintext correlation power analysis', in *2014 Tenth International Conference on Computational Intelligence and Security*, IEEE, November, pp.494–498.

Islam, N., Rahman, M.S., Mahmud, I., Sifat, M.N.A. and Cho, Y.Z. (2022) 'A blockchain-enabled distributed advanced metering infrastructure secure communication (BC-AMI)', *Applied Sciences*, Vol. 12, No. 14, p.7274.

Jain, A., Kohli, V. and Mishra, G. (2020) *Deep Learning Based Differential Distinguisher for Lightweight Cipher Present*, Cryptology ePrint Archive, DOI: 10.1109/ACCESS.2019.2939947.

Johnson, R. and Zhang, T. (2016) 'Supervised and semi-supervised text categorization using LSTM for region embeddings', in *International Conference on Machine Learning*, PMLR, June, pp.526–534.

Kalinin, N. and Skvortsov, N. (2021) 'Response to cybersecurity threats of informational infrastructure based on conceptual models', in *International Conference on Data Analytics and Management in Data Intensive Domains*, October, pp.19–35, Springer International Publishing, Cham.

Kandi, M.A., Kouicem, D.E., Doudou, M., Lakhlef, H., Bouabdallah, A. and Challal, Y. (2022) 'A decentralized blockchain-based key management protocol for heterogeneous and dynamic IoT devices', *Computer Communications*, Vol. 191, No. 6, pp.11–25.

Kanter, K.K. (2002) 'Secure exchange of information by synchronization of neural', *Europhysics Letters*, Vol. 141, No. 57, No. 57.

Keedwell, A.D. and Dénes, J. (2015) *Latin Squares and their Applications*, Elsevier, The Netherlands.

Kim, H., Lim, S., Baksi, A., Kim, D., Yoon, S., Jang, K. and Seo, H. (2023a) *Quantum Artificial Intelligence on Cryptanalysis*, Cryptology ePrint Archive.

Kim, H., Lim, S., Kang, Y., Kim, W., Kim, D., Yoon, S. and Seo, H. (2023b) 'Deep-learning-based cryptanalysis of lightweight block ciphers revisited', *Entropy*, Vol. 25, No. 7, p.986.

Koblitz, N. (1994) *A Course in Number Theory and Cryptography*, Vol. 114, Springer Science & Business Media, USA.

Kumari, K., Singh, J.P., Dwivedi, Y.K. and Rana, N.P. (2021) 'Multi-modal aggression identification using convolutional neural network and binary particle swarm optimization', *Future Generation Computer Systems*, Vol. 118, No. 9, pp.187–197.

Lagerhjelm, L. (2018) *Extracting Information from Encrypted Data using Deep Neural Networks*, Master Thesis, January, UMEA Univ., Dept. of Applied Physics and Electronics, Sweden.

Laywine, C.F. and Mullen, G.L. (1998) *Discrete Mathematics using Latin Squares*, Vol. 49, John Wiley & Sons, USA.

Li, Y. and Zhang, S. (2022) *Statistical Analysis. In Applied Research Methods in Urban and Regional Planning*, pp.109–148, Springer International Publishing, Cham.

Liao, M., Zheng, S., Pan, S., Lu, D., He, W., Situ, G. and Peng, X. (2021) 'Deep-learning-based ciphertext-only attack on optical double random phase encryption', *Opto-Electronic Advances*, Vol. 4, No. 5, pp.200016–200011.

Liu, Y., Zhang, C., Yan, Y., Zhou, X., Tian, Z. and Zhang, J. (2022) 'A semi-centralized trust management model based on blockchain for data exchange in IoT system', *IEEE Transactions on Services Computing*, Vol. 16, No. 2, pp.858–871.

Livieris, I.E., Kiriakidou, N., Stavroyiannis, S. and Pintelas, P. (2021) 'An advanced CNN-LSTM model for cryptocurrency forecasting', *Electronics*, Vol. 10, No. 3, p.287.

Luo, X. and Chen, Z. (2020) 'English text quality analysis based on recurrent neural network and semantic segmentation', *Future Generation Computer Systems*, November, Vol. 112, pp.507–511.

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

Munir, N., Khan, M., Hussain, I. and Alanazi, A.S. (2022) 'Cryptanalysis of encryption scheme based on compound coupled logistic map and anti-codifying technique for secure data transmission', *Optik*, Vol. 267, No. 4, p.169628.

Nakano, K. and Suzuki, H. (2022) 'Known-plaintext attack-based analysis of double random phase encoding using multiple known plaintext-ciphertext pairs', *Applied Optics*, Vol. 61, No. 30, pp.9010–9019.

Nakano, K., Takeda, M., Suzuki, H. and Yamaguchi, M. (2014) 'Security analysis of phase-only DRPE based on known-plaintext attack using multiple known plaintext – ciphertext pairs', *Applied Optics*, Vol. 53, No. 28, pp.6435–6443.

NIST1 (2022) *The National Institute of Standards and Technology (NIST)* [online] https://csrc.nist.gov/glossary/term/cryptography (accessed 22 October 2022).

NIST2 (2022) *NIST Special Publication 1800-21* [online] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1800-21.pdf (accessed 22 October 2022).

NIST3 (2022) *William C. Barker 'NIST Special Publication 800-59'* [online] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-59.pdf (accessed 22 October 2022).

NIST4 (2022) *The National Institute of Standards and Technology (NIST)* [online] https://csrc.nist.gov/glossary/term/cryptanalysis (accessed 22 October 2022).

NIST5 (2022) *The National Institute of Standards and Technology (NIST), NIST Special Publication 800-57 Part 2 Revision 1* [online] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf (accessed 22 October 2022).

Rao, J. and Cui, Z. (2022) 'Chosen plaintext combined attack against SM4 algorithm', *Applied Sciences*, Vol. 12, No. 18, p.9349.

Rao, K.K. (2009) 'Cryptanalysis of a Feistel type block cipher by feed', *International Journal of Soft*, January, pp.131–135.

Rundo, F. (2020) 'Deep LSTM with dynamic time warping processing framework: a novel advanced algorithm with biosensor system for an efficient car-driver recognition', *Electronics*, Vol. 9, No. 4, p.616.

Security (2022) *Purple Sec: Cyber Security Statistics, The Ultimate List Of Stats Data and Trends For 2022* [online] https://stefanini.com/en/insights/articles/cyber-security-statistics-for-2022-data-and-trends (accessed 23 October 2022).

Shao, J.Y. (1992) 'A formula for the number of Latin squares', *Discrete Mathematics*, Vol. 110, Nos. 1–3, pp.293–296.

Shin, H.S., Kwon, H.Y. and Ryu, S.J. (2020) 'A new text classification model based on contrastive word embedding for detecting cybersecurity intelligence in twitter', *Electronics*, Vol. 9, No. 9, p.1527.

Siegenthaler (1985) 'Decrypting a class of stream ciphers using ciphertext only', *IEEE Transactions on computers*, Vol. 100, No. 1, pp.81–85.

Stallings, W. (2019) *Information Privacy Engineering and Privacy by Design*, Addison-Wesley Professional, USA.

Stennikov, V., Barakhtenko, E., Mayorov, G., Sokolov, D. and Zhou, B. (2022) 'Coordinated management of centralized and distributed generation in an integrated energy system using a multi-agent approach', *Applied Energy*, March, Vol. 309, p.118487.

Tan, L., Yu, K., Ming, F., Cheng, X. and Srivastava, G. (2021) 'Secure and resilient artificial intelligence of things: a HoneyNet approach for threat detection and situational awareness', *IEEE Consumer Electronics Magazine*, Vol. 11, No. 3, pp.69–78.

Tolba, Z., Derdour, M., Ferrag, M.A., Muyeen, S.M. and Benbouzid, M. (2022) 'Automated deep learning BLACK-BOX attack for multimedia P-BOX security assessment', *IEEE Access*, September, Vol. 10, pp.94019–94039.

Van Lint, J. and Wilson, R.M. (1992) *A Course in Combinatorics*, Cambridge Univ. Press, New York.

Wade, M.I. and Gill, T. (2022) 'The Iso-ElGamal cryptographic scheme', in *2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, IEEE, June, pp.1–8.

Wan, Z., Wang, G., Yang, Y. and Shi, S. (2014) 'SKM: Scalable key management for advanced metering infrastructure in smart grids', *IEEE Transactions on Industrial Electronics*, Vol. 61, No. 12, pp.7055–7066.

WEB1 (2022a) *Alexandre Pelletier, Centralization vs. Decentralization: Why You Need A Hybrid Model More Now Than Ever* [online] https://perkuto.com/blog/centralization-vs-decentralization-why-you-need-a-hybrid-model-more-now-than-ever/ (accessed 23 October 2022).

WEB1 (2022b) *The Online Honeynet Project* [online] https://www.honeynet.org/ (accessed 31 August 2022).

Xiao, Y. (2019) 'Neural cryptanalysis: metrics, methodology, and applications in CPS ciphers', *2019 IEEE Conference on Dependable and Secure Computing (DSC).*