



Neural-driven dynamic key generation: A deep reinforcement learning and topological mathematics based framework for secure file sharing

R. Kavin Kumar^a , K. Supriya^a, A. Swetha Priya^a, Kamatchi S. ^a ,
Jitendra Bahadur^a, Dong-Won Kang^b

^a Department of Electronics and Communication Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Bengaluru, 560035, Karnataka, India

^b Department of Energy Systems Engineering, Chung-Ang University, 84 Heukseok-ro, Dongjak-gu, Seoul, 06974, Republic of Korea

ARTICLE INFO

Keywords:

Dynamic key generation
Deep learning
Security framework
Neural networks
Knot theory
Encryption and decryption techniques

ABSTRACT

The growing prevalence of data breaches in digital communication and platforms necessitates advanced mechanisms to secure adaptive file exchange systems. Traditional file-sharing methods reliant on static keys face challenges in maintaining security and adaptability in dynamic environments, which are increasingly susceptible to modern cyber threats. To address these challenges, a distinct computational framework was developed by integrating neural networks and deep reinforcement learning (DRL) to build a cryptographically secure pseudorandom number generator (CSPRNG) used to derive reliable and dynamic session-based keys, and algebraic knot theory-based topological transformations to improve cryptographic resilience and prevent key recovery in case of interception. This approach enables the framework to regularly update session-based keys to bypass vulnerabilities inherent in static systems while applying complex mathematical structures that advance resilience against cyberattacks. The framework's efficacy has been validated through multiple experiments, including entropy and randomness tests, hamming distance analysis, and performance benchmarks that confirm both unpredictability and negligible runtime overhead. These results demonstrate that the framework achieves strong security properties while maintaining practical efficiency. Although the primary focus is on secure file sharing, the results confirm that the framework is efficient, adaptable, and practical within this domain, showing promise as a foundation for future exploration in other secure communication settings such as mobile, distributed environments and lightweight platforms.

1. Introduction

As digital communication evolves and new techniques and technology emerge, secure file-sharing has become an increasingly pressing concern. Companies and individuals depend significantly on digital platforms to share sensitive or personal information, making them susceptible to cyberattacks. The increase in data breaches and unauthorized access to sensitive information reflects the inadequacies of traditional security measures and highlights the need for more flexible and dynamic solutions. As the dependence on cloud-based services and other digital platforms grows, the threat of cyberattacks continues to increase, pushing

* Corresponding author.

E-mail address: s_kamatchi@blr.amrita.edu (Kamatchi S.).

<https://doi.org/10.1016/j.rico.2026.100744>

Received 6 March 2026; Received in revised form 15 April 2026; Accepted 21 May 2026

Available online 10 June 2026

2666-7207/© 2026 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

the community to explore various innovative technologies to improve security. The application of Artificial Intelligence (AI) and Machine Learning (ML) and its integration with traditional cryptography is one promising area of research. Beyond traditional approaches, machine learning techniques such as neural networks [1] and generative adversarial networks (GANs) have also been explored for cryptographic applications. A GAN consists of two models trained in competition: a generator that produces candidate outputs and a discriminator that evaluates them. Through this adversarial training process, the generator learns to produce outputs that closely resemble true randomness. This ability has led to several proposals where GANs are used to generate encryption keys or secure pseudorandom sequences. While promising in terms of unpredictability, these methods often require significant computational resources and careful tuning, which can limit their practicality in lightweight file sharing systems. In this study, deep reinforcement learning (DRL) is preferred as it provides adaptability and high entropy while remaining efficient enough for real time key generation. Deep Reinforcement Learning (DRL), a type of hybrid AI technique that combines deep learning and reinforcement learning, teaches machines to take actions based on feedback, which is received in the form of a reward or a punishment. This hybrid technique shows potential in cryptographic systems, particularly in generating random numbers used as the seed for deriving dynamic keys. Most existing systems still rely on static keys generated through PRNG, which leaves them vulnerable to attacks. A PRNG follows a deterministic process, which means that once its initial seed is known the entire sequence of values can be reproduced. While this behavior is acceptable in simulations or numerical applications, it creates a weakness in cryptography because an attacker who uncovers the seed can predict every future key. Algorithms such as the Linear Congruential Generator and Mersenne Twister illustrate this problem. They are efficient and widely used but not resistant to state recovery or prediction attacks. Cryptographically secure PRNGs (CSPRNGs) were introduced to overcome these weaknesses by ensuring that even partial knowledge of the sequence does not reveal the rest. The framework developed in this work builds on this principle by integrating a DRL-based generator that produces high entropy and session specific keys, making them unpredictable and unique for each communication session. The most common and industry standard for encryption today is the Advanced Encryption Standard (AES) algorithm, known for its efficiency and robust security, especially in encrypting large amounts of data. AES has been in use since 2001, and its symmetric-key predecessor, Data Encryption Standard (DES), had been in use since the 1970s. Rivest-Shamir-Adleman (RSA) [2], an asymmetric-key algorithm that employs a public-private key pair for encryption and decryption, operates on a fundamentally different cryptographic paradigm and provides strong security for data transmission independently of AES. While AES has proven to be incredibly secure, it has its limitations. RSA relies on the difficulty of factoring large composite numbers but shows vulnerabilities to certain types of attacks. AES and RSA suffer from inherent susceptibilities due to static keys and their management, highlighting the need for dynamic and adaptive cryptographic [3,4] solutions. This study demonstrates a unique approach to integrating DRL into traditional cryptographic methods while introducing an additional layer of mathematical security through algebraic knot theory. Knot theory, a branch of mathematics that studies the properties and behavior of knots and curves, introduces complex transformations that can significantly enhance the security of cryptographic keys [5]. Applying these transformations to the AES [6] key before encryption helps the system create a unique mathematical structure that makes it harder to decipher. This combination of dynamic key generation and key transformation creates a more robust and resilient security framework for file sharing [7–10]. This approach offers several significant advantages. The random number generation by DRL ensures that the encryption keys are distinct, thereby lessening the risk of compromise, and incorporating knot-theory transformations enhances security, rendering the keys more difficult to predict or reverse-engineer. By integrating neural networks, deep reinforcement learning and knot theory, this study introduces an unexplored framework for an advanced secure file-sharing system, combining features such as session-based file sharing and mitigation of key storage and management to overcome existing problems, contributing to the development of strengthened and more reliable security systems. It is flexible and scalable, allowing for adaptation across a diverse range of cryptographic applications, thus broadening its potential beyond file-sharing.

To clearly distinguish the proposed framework from existing cryptographic and adaptive secure file sharing mechanisms, its novel contributions are threefold. First, unlike traditional deterministic algorithms that rely on static seeds, the proposed DRL-based CSPRNG dynamically adjusts its output based on session context and entropy-driven rewards, ensuring high-entropy, session-specific key generation. Second, while conventional encryption standards utilize fixed mathematical transformations strictly within the cipher's data path, the developed framework leverages algebraic knot theory to introduce an independent, reversible key-hardening layer prior to transmission. This topological obfuscation requires an attacker to determine the exact Reidemeister move sequence and polynomial parameters to recover the key, substantially increasing the computational workload even if the transport wrapper is compromised. Finally, whereas many existing protocols rely on static key storage, the current framework ensures forward secrecy by generating, utilizing, and discarding keys on a strict per-session basis, thereby minimizing the vulnerability window. A detailed quantitative comparison of these differences against standard mechanisms, such as RSA-KEM and ECDHE, is provided in the subsequent sections and supplementary material.

By integrating neural networks, deep reinforcement learning, and knot theory, this study introduces an unexplored framework for an advanced secure file sharing system. It combines features such as session-based file sharing and the mitigation of key storage management to overcome existing vulnerabilities, contributing to the development of strengthened and more reliable security systems. The methodology is flexible and scalable, allowing for adaptation across a diverse range of cryptographic applications, thus broadening its potential beyond file sharing.

2. Relative work

Ding et al. [11] propose a deep learning-based stream cipher generator inspired by GAN to automatically generate private keys, eliminating the disadvantages of hand-designed security key generators. The proposed DeepKeyGen transforms source domain

images into a target transformation domain of security key style, achieving high randomness, a large key space (2196,608), and sensitivity for strong encryption. Evaluations on diversified medical imaging data, such as X-ray, ultrasound, and MRI, confirm its ability to encrypt and securely decrypt multi-modal data while resisting brute force, imitation learning, and differential attacks. The system ensures high statistical randomness, achieving NPCR > 99.5% and UACI > 23%, and removes statistical patterns in plaintexts as verified by entropy and histogram analyses. However, its computational intensity and reliance on secure transformation domains limit its deployment on resource-constrained devices. Despite these limitations, DeepKeyGen represents a significant advancement in combining machine learning with cryptographic systems to protect sensitive medical data. Chunka, C. et al. [12] a new method to enhance encryption security is proposed: GACSAVK, a Genetic Algorithm-based Circular Shifting Automatic Variable Key. Genetic algorithms are utilized to create an initial key from a population of potential keys, selecting the fittest using a fitness function. Dynamic keys are then generated through a circular shifting technique integrated with an automatic variable key mechanism for each data block. Experimental results show that this scheme surpasses traditional methods like AVK and RC4 in terms of randomness, Average Hamming Distance (AHD), and Standard Deviation (SD) between successive keys. Unlike AVK, GACSAVK prevents key repetition even with identical plaintext blocks, reducing risks associated with statistical and ciphertext-only attacks. Randomness testing [13] using the NIST statistical suite validates its ability to generate unpredictable and secure keys. Despite its effectiveness, the method's reliance on computationally intensive GA-based selection and shifting mechanisms may hinder efficiency in resource-constrained environments. Nevertheless, GACSAVK represents a significant advancement in cryptographic security, making it a robust solution for secure data transmission [2]. Yu et al. [14] present an improved RSA variant encryption system with optimized performance for resource-constrained environments. The mechanism utilizes bio-signal-based True Prime Random Number Generators (TPRNG) [15] from photoplethysmogram (PPG) sensors to generate unpredictable prime numbers for security key creation. Unlike traditional RSA, which develops large 2048-bit keys, this system develops compact 32-bit keys that are dynamically discarded and regenerated after each use, reducing the risk of side-channel attacks. The TPRNG, validated for high entropy through NIST randomness tests, ensures strong security. This approach significantly reduces computational and memory demands, making the system suitable for IoT environments while providing sufficient safety for scenarios that do not require stringent encryption. However, the smaller key size limits its application in high-security settings. Successful hardware implementation on an FPGA demonstrates the system's efficiency in encryption and decryption, proving its practicality for real-time communication in IoT and small wireless devices. Ullah et al. [16] propose a deep learning-based hashing algorithm developed to mitigate vulnerabilities such as collision, birthday, and meet-in-the-middle attacks found in existing systems. The method employs dynamic secret key generation and an innovative pattern-matching mechanism to efficiently secure the input data of variable lengths. The algorithm divides input messages into pairs of four characters, addressing conflicts and incorporating dynamic keys created through forward and backward propagation rules. Collision-free hash values are generated by combining these keys with binary transformations of the inputs. Experimental results highlight superior performance in sensitivity, speed, and collision resistance compared to established hashing methods like MD5 and SHA-256. Despite its efficiency, the system's reliance on deep learning models increases computational complexity, making it potentially resource-intensive for straightforward applications. Nonetheless, it offers a robust solution for secure data handling [17] across various communication scenarios. Wu, X. et al. [18] provide a comprehensive review of pseudorandom number generators (PRNGs) that leverage neural networks, categorizing existing approaches into recurrent neural network (RNN) based generators, generative adversarial network (GAN) based designs, and reinforcement learning (RL) based schemes. RNN and LSTM models exploit sequential dependencies to generate long streams with improved statistical properties, GANs employ adversarial training to approximate true randomness distributions more closely, and RL based methods adaptively refine generator parameters [19] through policy optimization to improve unpredictability. The survey emphasizes the advantages of neural models, including nonlinear representation capacity, adaptability across tasks, and scalability through parallel computing, while also identifying limitations such as reproducibility issues, high computational demands, sensitivity to hyperparameters, and the lack of standardized security benchmarks. Unlike individual proposals that introduce specific architectures, this work systematizes the field by unifying design principles, summarizing evaluation strategies such as NIST statistical testing, and comparing algorithmic characteristics across categories. The authors conclude that neural network based PRNGs hold strong potential for cryptographic and simulation applications, but further progress requires more rigorous security proofs, common evaluation frameworks, and efficiency improvements to ensure practical deployment. Xiao, Q. et al. [20] present a survey on physical-layer key generation (PKG) as a security solution for NextG wireless networks, reviewing recent advances, open issues, and future opportunities. The paper systematically examines the fundamental principles of PKG based on wireless channel reciprocity and randomness, and organizes existing work across scenarios such as MIMO, millimeter-wave, and reconfigurable intelligent surface (RIS) assisted systems. It highlights that PKG offers lightweight, information-theoretic security that can complement or replace conventional cryptography in resource-constrained and latency-sensitive settings. The authors also identify challenges [21] including channel asymmetry in FDD systems, limited key generation rates in high-mobility environments, vulnerability to active attacks, and difficulties in standardizing evaluation metrics. Unlike individual technical proposals, this survey unifies the state of the art by comparing methodologies, key performance indicators, and application domains, while also discussing integration with emerging NextG technologies such as edge intelligence and semantic communications. The study concludes that PKG represents a promising approach to secure future wireless networks, but practical adoption will depend on resolving scalability, interoperability, and robustness issues across diverse deployment environments. Sconza et al. [22] propose an innovative cryptographic key-exchange technique based on knot theory, considering the semigroup of oriented knots under connected sum operations. This key-exchange [9,23,24] protocol differs from traditional group-based cryptosystems and Diffie–Hellman-based cryptosystems, using a semigroup action, assuming the hardness of computing decompositions of connected sums of knots for security reasons. The protocol establishes a shared secret among the participants, having finite type invariants that are computable efficiently and robust against connected sum-based attacks. This paper

details the cryptographic advantage of semigroups over groups: immunity to standard attacks based on the inverses, such as Pollard's rho algorithm, and it ensures the choice of parameters [25,26] like the number of crossings such that it is computationally feasible yet secure against brute-force attacks. While it provides an innovative approach, its efficiency is challenged by the computation overhead for finite-type invariant calculations, which scale polynomials with crossing numbers. Therefore, it is recommended that future work improve the computations and look into other invariants that might further enhance efficiency without trading security. This protocol marks a significant step toward topological constructs within cryptography and finding a novel approach to overcoming post-quantum security challenges. SreeDeepika, M., et al. [1,7–9,19,23,27–30] discuss the practical implementation of the AES algorithm for securing textual and image data. The authors employed MATLAB to implement AES, demonstrating its effectiveness in encrypting and decrypting diverse data types, including characters, text strings, and color images. The encryption process transforms plaintext into unreadable ciphertext through multiple rounds of operations such as byte substitution, row shifting, column mixing, and adding a round key, while decryption reverses these steps. A consistent 128-bit key and block size were used across experiments, highlighting AES's capability to maintain data integrity without distortion during transmission. For image encryption, the RGB components were processed separately, enabling the transformation of each 4×4 data block. The results confirmed AES's efficiency, with decrypted outputs matching the original inputs, ensuring security against unauthorized access. However, the study was limited to AES-128 and MATLAB without investigating hardware acceleration optimizations or comparisons with other encryption schemes. This work underscores AES's robustness in secure communications, with potential applications in sensitive sectors like military and banking, while providing opportunities for future research on efficiency enhancements and higher-bit security key implementations. T. T. Nguyen, et al. [28] provide a comprehensive review of DRL techniques developed to address complex and dynamic challenges in cyber security. This paper emphasizes the need for adaptive and predictive defensive mechanisms in the face of sophisticated cyber threats, especially in IoT, CPS, and autonomous systems. Integrating deep learning with reinforcement learning enables DRL methods to deal with high-dimensional, sequential decision-making problems, which makes them suitable for detecting and mitigating advanced threats. The paper categorizes DRL applications into CPS defence, autonomous intrusion detection, and game theory-based approaches for strategic cybersecurity scenarios. Among its vital contributions, it focuses on A3C and DQN real-time CPS monitoring, distributed strategies for DDoS attacks, and adversarial models to combat deception and spoofing attacks. This said, the authors did outline challenges that remain unachieved among those set forward, such as the gap from simulations toward applications into reality and the underutilized potential of model-based DRL combined with the lack of dynamic environments to properly train such environments. Conclusion Paper Conclusion The final paper shall also address future directions for more in-depth research, particularly advocating the need for more hybrid methods of DRL and even proper modeling of the cyber system. Oh, S. H., et al. [29] discuss the application of deep reinforcement learning (DRL) in cybersecurity to simulate and mitigate complex cyber threats. This study highlights the inadequacies of traditional security frameworks and emphasizes the adaptability of machine learning techniques, in particular, DRL, in dynamic settings. The researchers established a deep reinforcement learning framework in the simulated environment designed for cyber-attacks, using algorithms such as deep Q-network, actor-critic, and proximal policy optimization (PPO) versus Q-learning. The experiments showed that the actor-critic algorithm achieved the best success rate and efficiency, signifying its ability to rapidly adapt and learn optimal policies. This research pushes the state of the art using the MITRE ATTACK framework for realistic scenario modeling and provides insights into algorithm-specific advantages in cybersecurity applications. The study emphasizes DRL's potential to enhance cybersecurity defences and spur innovation in adaptive threat response systems. Future work will focus on the challenges of DRL being resource-intensive and the requirement for robust transfer learning frameworks for real-world applicability. To better define the architectural contributions of the proposed methodology, it is compared against two industry standard benchmarks: Transport Layer Security (TLS) and the Signal Protocol. As implemented in the socket based transmission modules, TLS provides essential point to point encryption and server authentication using X.509 certificates. However, TLS security is primarily concentrated at the transport layer. The framework presented in this study complements TLS by adding a pre-transmission hardening layer where the session key is topologically transformed via Reidemeister moves. This provides a secondary layer of protection against transport layer vulnerabilities such as compromised Certificate Authorities. In contrast, the Signal Protocol is designed for secure, asynchronous end to end messaging. It utilizes the Double Ratchet algorithm to achieve perfect forward secrecy for individual messages. While the Signal Protocol is highly effective for low latency communication, its overhead can be significant for large scale file transfers. The proposed protocol is optimized for high throughput file sharing by utilizing a DRL based CSPRNG to generate ephemeral, high entropy session keys that are then wrapped via RSA-OAEP. By combining the rapid encryption of AES-GCM with a knot theory based key disguise, the system achieves a balance between the cryptographic resilience of the Signal Protocol and the computational efficiency required for multi-gigabyte data sharing.

3. Background

3.1. Knots

A Knot is a closed curve embedded in a three-dimensional Euclidean space R^3 , which is interlaced in a manner that cannot be undone without cutting the string. Two or more knots together are called a link. A mathematical knot is different from the usual idea of a knot, which is a piece of string with free ends. It represents a specific arrangement of the rope where the string is twisted or interlaced without being torn, and it cannot be separated through simple deformations like stretching or bending. The vital characteristic of a knot is its ability to maintain its structure under various topological manipulations, and its properties remain independent of the material of the rope. Two knots or links are considered equivalent if one can be smoothly deformed into the other without cutting or tearing, using specific transformations known as Reidemeister moves. Alternatively, they are equivalent if there exists a homeomorphism in R^3 that maps the image of the first knot onto the second. Determining if two knots are equivalent poses

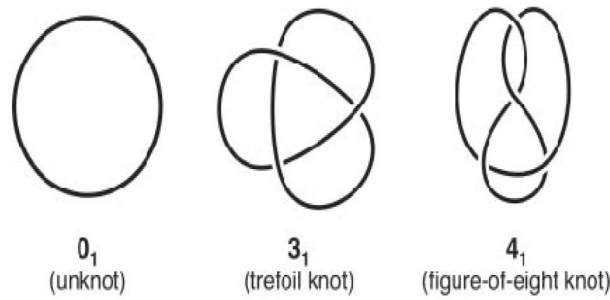


Fig. 1. Simplest knots.

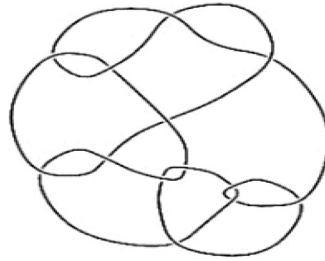


Fig. 2. Complex knot.

a substantial challenge, and much of knot theory is devoted to developing techniques to aid in addressing this question. Knots that are equivalent to polygonal paths in three-dimensional space are called tame. All other knots are known as wild. Fig. 1 illustrates the simplest possible knots and Fig. 2 illustrates a complex knot.

Knots are classified based on several vital properties that help differentiate between their structure. The most fundamental property is the crossing number, which denotes the minimum number of crossings needed in a diagram to represent the knot. This number serves as a measure of the knot's complexity. Knot types are also assessed for their triviality; that is, they can be classified as trivial or non-trivial. A trivial knot or an unknot is a loop that can be untangled to an unknotted form, whereas non-trivial knots are cut to simplify and unknot. Additionally, knots can be categorized as prime or composite. A prime knot cannot be decomposed into fundamental knots, while a composite knot is formed by combining two or more prime knots. To further classify knots, the knot invariance property plays a crucial role. The properties of knots that remain unchanged under deformations through Reidemeister moves are known as knot invariants. These criteria for classification provide a framework to analyze knots in theoretical and applied contexts. The knots are notated as xy , with x denoting the number of crossings and y indicating the index of the unique knot with x number of crossings. E.g. 8_6 refers to a specific knot with 8 crossings and the 6th unique knot with 8 crossings. Over 1.7 million prime knots have been discovered with up to 16 crossings. The topology of the keys can be represented in the form of a polynomial, this study uses the HOMFLY polynomial which is a mathematical invariant of knots in a three-dimensional space. Eq. (1) represents a HOMFLY polynomial.

$$(V^{-4} - 2 + 2v^2) + (v^4 - 2v^{-2} - 3 + v^2)z^2 + (-v^{-2} - 1)z^4 \quad (1)$$

3.2. Knot theory

Knot theory is the branch of topology that studies the mathematical properties of knots. It concentrates on the classification and analysis of knots by studying the closed curves in three-dimensional space and their probable deformations. The field explores the characteristics of knots, intending to classify and understand the structures that arise when a string is twisted, interlaced, or entangled in ways that cannot be undone without cutting the formed loop. The primary objective of knot theory is to determine the equivalency between two knots, i.e. the knots can be transformed into one another without breaking the loop or allowing it to pass through itself. The knots are formalized through three fundamental transformations: twisting, poking and sliding. These transformations, known as the Reidemeister moves, allows for the manipulation of the knot's diagram while maintaining its topological properties. If a knot can be transformed into another using a finite sequence of Reidemeister moves, they are considered equivalent. Fig. 3 illustrates the transformations in the Reidemeister moves with type I, II and III illustrating twisting, poking and sliding respectively.

Beyond theoretical exploration, knot theory and its underlying principles have applications in various domains. This study explores the principles of knot theory and its application in cybersecurity to add a mathematical layer of security to the keys. A generalized explanation of how knot theory is integrated into cryptography has been provided in the supplementary information (Section 1) and a detailed comparison between knot-based transformation and conventional transformation across various aspects is provided in the supplementary information (Section 2).

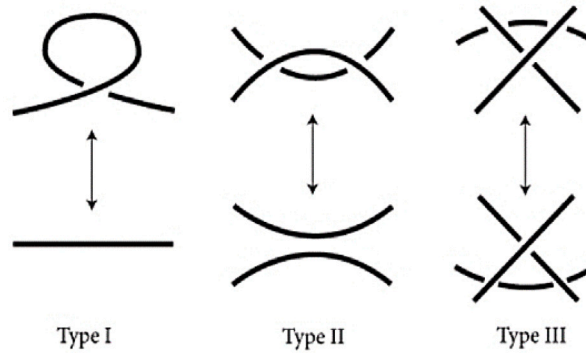


Fig. 3. Reidemeister moves.

4. Methodology

In this section, the proposed framework for the file-sharing system is outlined. The primary aim is to enhance the system by adding additional layers of security while making it dynamic and sessional to mitigate the vulnerabilities that arise from static key generation, storage and management.

4.1. Flow of the proposed framework

The proposed framework's workflow is illustrated in the block diagrams below. Fig. 5 depicts the transmission process between the sender and receiver, while Fig. 6 outlines the decryption and processing workflow at the receiver's end following transmission. This workflow can be divided into two main sections:

1. The transmission phase between sender and receiver and
2. The post-transmission phase at the receiver's end

To clearly illustrate the interaction between the learning components and the cryptographic pipeline, a comprehensive architecture diagram is presented in Fig. 4. The workflow begins with the Neural Network Module, where a lightweight Convolutional Neural Network (CNN) extracts embedding features from the input state. These features are concatenated with handcrafted statistical descriptors to form the environmental state space. This state is fed into the Deep Reinforcement Learning (DRL) Agent, which utilizes a Proximal Policy Optimization (PPO) actor-critic architecture to generate a 32-byte high-entropy stream. This stream acts as the seed for the Cryptographically Secure Pseudorandom Number Generator (CSPRNG) process, utilizing PBKDF2 to derive the final 256-bit AES session key. Finally, the derived AES key is passed through the Topological Transformation Module, where HOMFLY-derived parameters dictate a sequence of Reidemeister moves to mathematically disguise the key prior to RSA wrapping and transmission.

4.1.1. Between the sender and the receiver

The process begins by establishing a connection created by a socket [8,31] network between the sender and receiver, enabling P2P data transmission and facilitating the transfer of files and encryption keys between the two parties. On the sender's side, an image is provided as input to the DRL-CNN hybrid model, which generates a data stream. This stream is hashed into a secure 32-byte hexadecimal cryptographic key using the PBKDF2 (Password-Based Key Derivation Function 2) algorithm. The files for sharing (a single file or a directory) are compressed using one of the three compression options: zlib, lzma, or bz2. The compressed file is then encrypted using the generated AES key and transmitted to the receiver. Prior to transmission, the AES key is transformed using a sequence of Reidemeister moves to create a transformed key. The transformed AES key is encrypted using the receiver's RSA public key. Upon receiving the encrypted file and transformed AES key, the original files can be retrieved by decrypting the key with its RSA private key following which the transmitted data can be decrypted and securely accessed.

4.1.2. Receiver's end

An RSA public and private key pair is generated using the extended Euclidean algorithm, and the public key is shared with the sender. On the sender's side, once the encryption process is completed using the public key, the encrypted transformed key and the transformation sequence are transmitted to the receiver. The RSA private key is employed to decrypt the encrypted transformed security key, allowing the Reidemeister moves to be reversed and retrieve the original AES key. Using this AES key, the receiver decrypts the file transmitted by the sender. Once decrypted, the original content is extracted and made accessible to the user, before the socket disconnects. In the event of a socket timeout, the session terminates and reconnects with a new key pair and dynamic key to ensure security during reconnection. A linear step-by-step breakdown of the methodology has been provided in the supplementary information.

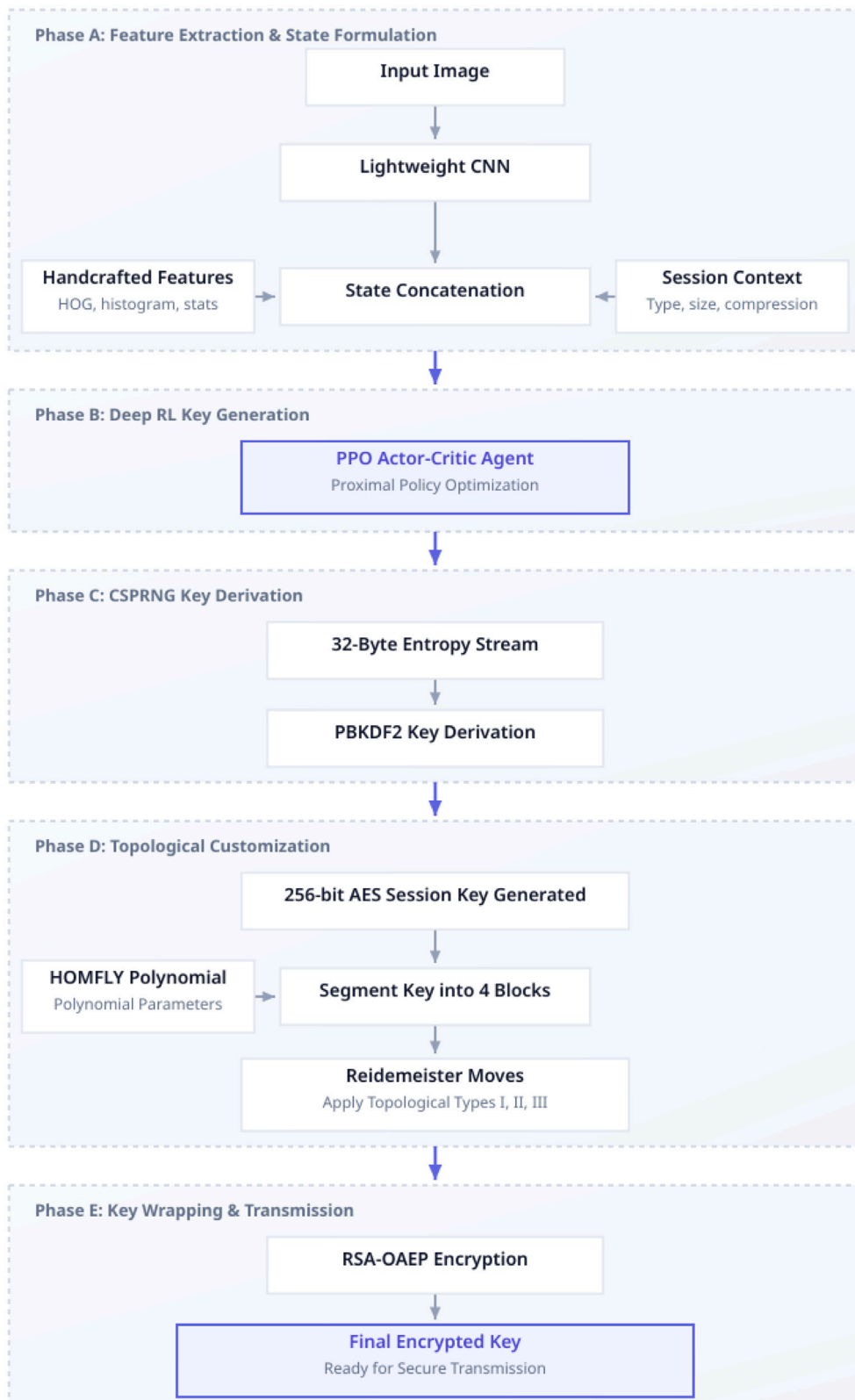


Fig. 4. Comprehensive architecture diagram illustrating the data flow and interaction between the CNN feature extractor, the DRL PPO agent, the CSPRNG key derivation process, and the knot-theory topological transformation module.

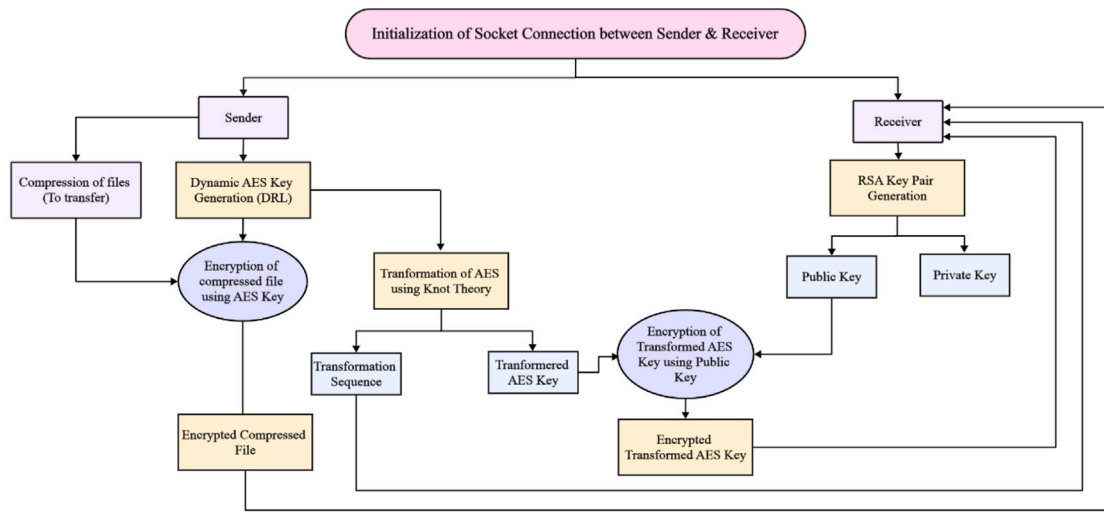


Fig. 5. Workflow of the framework between sender and receiver.

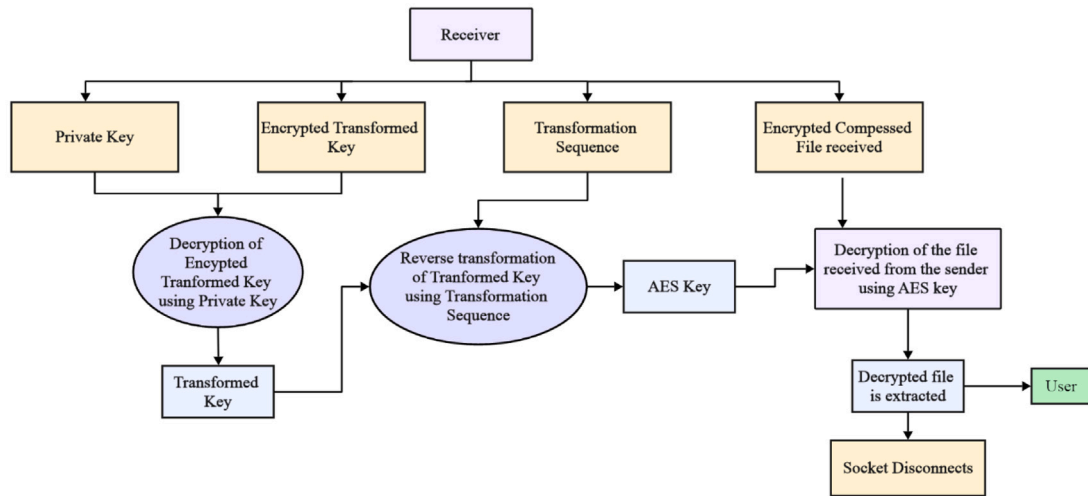


Fig. 6. Workflow of the framework on the receiver's end.

4.2. Dynamic key generation

The dynamic security-key generation process is an advanced approach to mitigating the vulnerability issues associated with static keys. The process is designed to generate true randomness in keys, thus improving security. The proposed key generation process employs a Proximal Policy Optimization (PPO) based actor-critic model. In this configuration, the actor generates candidate random sequences while the critic evaluates them against an entropy-driven reward function. This setup encourages the model to maximize unpredictability and avoid repeatable patterns, ensuring that the resulting output is suitable for cryptographic use. The system combines handcrafted features such as Grayscale Conversion, HOG Feature Extraction and Dimensionality Reduction with deep features derived from a convolutional neural network to create a thorough and robust representation of the input data. The actions of the agent are used iteratively for the generation of a random number set represented as a decimal byte array, with rewards calculated based on the entropy of the generated sequences, ensuring randomness. The model's learning process is optimized through advantage estimation, gradient clipping, and entropy regularization to balance exploration and exploitation. The model can be tweaked to generate random numbers of any key length.

The proposed dynamic key generation process is explicitly classified as a Cryptographically Secure Pseudorandom Number Generator (CSPRNG) rather than a True Random Number Generator (TRNG). A TRNG relies on sampling unpredictable, non-deterministic physical phenomena, such as thermal noise or hardware-based quantum variations, to generate raw entropy. In contrast, the

```

Generated Key: [87, 19, 237, 6, 155, 87, 14, 199, ... 248, 170, 242, 61, 132, 97]
Generated Key Entropy: 7.99428
Salting Key: 9b734a0dec6e6b78d38f3ef1e4308cf2cf68d3168ab8b8e12520d53e42b0db7
Derived AES Key: fff6fd2567106c4300882f00d631fbd7d5f412129dfdda23eb71417049c86519
Time Taken: 1.83 seconds

```

Fig. 7. Dynamically generated byte array key, hashed salt value and pbkdf2 derived AES key.

developed framework operates computationally, utilizing a Convolutional Neural Network (CNN) to extract environmental features and a Proximal Policy Optimization (PPO) based actor-critic model to generate random sequences. Because the generation process relies on algorithmic computations and defined state representations rather than physical entropy sources, it is fundamentally a pseudo-random process. However, the DRL agent is explicitly trained using an entropy-maximizing reward function that penalizes repeatable patterns and predictable n-grams. This deep non-linear transformation ensures that the generated sequences achieve near-ideal Shannon entropy and possess the high degree of computational unpredictability required to resist state recovery and prediction attacks. Consequently, while it is computational by design, the system fulfills the stringent statistical requirements of a CSPRNG, rendering it highly suitable for secure cryptographic session key derivation.

Since the framework in this study uses AES, a key derivation, primarily the PBKDF2 function, is leveraged by the system to generate the final hashed 256-bit security key used for file encryption. The random number set generated is the seed for the PBKDF2 function. The number set is then hashed into a 256-bit string used as the salt during the key derivation, and finally, a 256-bit security key is output. This ensures that the generated key and salt are always dynamic and cannot be interpreted by performing cyberattacks. Fig. 7 represents the generated cryptographically secure random number along with its entropy, hashed salt, derived final key and time taken.

The neural backbone of the framework consists of a lightweight CNN with three convolutional blocks (Conv → ReLU → MaxPool) followed by a dense projection layer. The CNN processes the input image to produce a compact embedding, which is concatenated with handcrafted features such as grayscale intensity histograms, HOG descriptors, and reduced statistical vectors, along with session context parameters including file type, size class, and compression flag. The concatenated feature vector forms the state representation for the DRL module. For policy learning, Proximal Policy Optimization (PPO) is employed in an actor-critic configuration. The actor generates one 8-bit symbol per step, and across 32 steps emits a 32-byte sequence that seeds the key derivation process, while the critic estimates state value for advantage computation. The reward function is entropy-driven, targeting approximately eight bits per byte, with penalties for repeated bytes and high-frequency n-grams. This design encourages high-entropy, unpredictable output suitable for session key material. The resulting 32-byte stream is processed through PBKDF2 to derive a 256-bit AES key, while the same stream is also hashed to produce a per-session salt. The combination ensures that each session key is unique, ephemeral, and never reused.

The reinforcement learning component enables adaptive key generation by optimizing for high entropy, low correlation, and cross-session unpredictability. Unlike fixed-function PRNGs, the DRL agent dynamically adjusts its output in response to input features and session context, ensuring that each session produces statistically independent material. Empirical results confirm that the generated sequences exhibit near-ideal entropy (Approx. 7.9966 bits/byte) and pass standard randomness batteries while maintaining large Hamming distances across sessions. These properties make the keys resistant to prediction and correlation, providing a stronger basis for PBKDF2-derived AES keys than conventional static or pseudo-random generators.

The model trains via an episodic reinforcement learning paradigm utilizing Proximal Policy Optimization (PPO). During each training episode, the actor network sequentially generates 8-bit symbols over 32 time steps to construct a continuous 32-byte stream.

Unlike traditional supervised learning models that rely on fixed and labeled datasets, the DRL agent learns dynamically through environmental interaction. The operational training set in this context consists of a diverse corpus of input images containing varying textures, contrasts, and structural complexities. These images are processed by the convolutional feature extractor to bootstrap the initial state representation for each episode. Exposing the agent to a highly variable input space ensures that the learned policy generalizes and avoids deterministic overfitting.

The optimization of the agent is governed by an entropy-centric reward and penalty formulation. The primary objective function rewards the agent proportionally to the Shannon entropy of the generated byte stream, driving the output toward an ideal 8 bits per byte. In contrast, to strictly prevent policy collapse into predictable patterns, a mathematical penalty function is applied. The agent incurs negative rewards, or penalties, for generating repeated consecutive bytes, low-variance subsequences, and high-frequency n-grams. Through this continuous reward and penalty structure, the PPO algorithm updates its policy using a clipped surrogate loss, ultimately training the agent to maximize cryptographic unpredictability across any given input state.

Mathematically, the total reward function $R(S)$ for a generated 32-byte sequence $S = (s_1, s_2, \dots, s_{32})$ is formulated as a composite function of Shannon entropy and specific structural penalties:

$$R(S) = \alpha \cdot H(S) - \beta \cdot P_{rep}(S) - \gamma \cdot P_{freq}(S) \quad (2)$$

Where:

- $H(S)$ represents the primary positive reward, calculated as the Shannon entropy of the sequence: $H(S) = -\sum p(x) \log_2 p(x)$, driving the distribution toward 8 bits per byte.

```
['0x57', '0x21', '0x70', '0x5b', '0x70', '0x7', '0x0', '0x96', '0x57', '0x8a', '0xf5',
'0xa1', '0xab', '0x31', '0x61', '0x4', '0x4f', '0x87', '0x7c', '0x9a', '0x8f', '0x50',
'0x2b', '0x52', '0x43', '0x3e', '0x9f', '0x41', '0xd8', '0xdf', '0x3', '0x43']
```

Fig. 8. Example of an original key bytes from the 32-byte AES key.

```
Segment 1:
['0x57', '0x21', '0x70', '0x5b', '0x70', '0x7', '0x0', '0x96']

Segment 2:
['0x57', '0x8a', '0xf5', '0xa1', '0xab', '0x31', '0x61', '0x4']

Segment 3:
['0x4f', '0x87', '0x7c', '0x9a', '0x8f', '0x50', '0x2b', '0x52']

Segment 4:
['0x43', '0x3e', '0x9f', '0x41', '0xd8', '0xdf', '0x3', '0x43']
```

Fig. 9. Example of a segmented 32-byte AES key into four 8-byte parts.

- $P_{rep}(S)$ is the consecutive repetition penalty, defined as $\sum_{i=1}^{31} \mathbb{I}(s_i = s_{i+1})$, where $\mathbb{I}$ is an indicator function that applies a negative weight whenever adjacent bytes are identical.
- $P_{freq}(S)$ is the n-gram frequency penalty, which applies a logarithmic scale penalty if any specific bigram or trigram exceeds an expected uniform distribution threshold.

The hyperparameters α , β , and γ are scaling coefficients used to balance the exploration space, ensuring the PPO agent prioritizes overall entropy while strictly avoiding local optima containing predictable byte clusters.

4.3. File compression and encryptiondynamic

Once the AES key is generated, the files are prepped to be compressed and encrypted. The files can be compressed using one of three compression algorithms: zlib, lzma, and bz2. Each algorithm offers a unique balance between speed and compression efficiency; zlib is faster but provides moderate compression, lzma achieves higher compression ratios at the cost of speed, and bz2 serves as a balanced option in terms of both speed and compression effectiveness. A range of file types, such as .txt, .jpg, and .pdf are supported for compression. Compression is possible on both single files as well as directories. For directory compression, recursive compression of all supported files within the directory structure takes place. Upon completion of the compression process, the files are encrypted using the 32-byte dynamically generated key. It utilizes the Advanced Encryption Standard (AES) with a 256-bit key in Galois/Counter Mode (GCM), an algorithm known for its robustness and efficiency in authenticated encryption. Utilization of this GCM mode generates an authentication tag that ensures data integrity by checking for any data tampering or interception. To further enhance data integrity, a Hash-based Message Authentication Code (HMAC) using the SHA-384 hash function over both the initialization vector (IV) and the encrypted data is employed. It also generates a unique random IV for each encryption session, which is critical for maintaining the security properties of AES-GCM. The framework efficiently handles large datasets without compromising performance by processing the file in chunks. This encrypted file is transmitted over to the receiver.

4.4. Key transformation

One of the most vulnerable intersections during file sharing is transmitting the AES key to the receiver for file decryption. In this proposed methodology, a theoretical workflow replicates the process of the security key transformations using Reidemeister moves. The dynamically generated original 32-byte AES key is broken down to 32 individual bytes as represented in Fig. 8. These bytes are subsequently organized into four segments, each consisting of eight bytes, with each segment representing a distinct part of the key, as illustrated in Fig. 9. A HOMFLY polynomial, as shown in Fig. 10, is provided as input. Each term in the polynomial is broken down into three parts: the coefficient, the power of v , and the power of z . These parts, represented in Fig. 11 as individual terms, are parsed and stored in a dictionary, with each unique combination of powers of v and z serving as keys, and the coefficients as values. This dictionary will be used later when applying transformations to the segments of the AES key.

Each segment is further modified depending on the specified move using a set of functions, where each function represents a Reidemeister move. These moves are used as the transformation sequence to modify the AES key. The Reidemeister moves are modifiable based on specific security requirements. The Reidemeister moves are mapped as follows in this study:

1. The R1 move adjusts the coefficient of a specific term in the polynomial based on its current value. If the absolute value of the coefficient is greater than 5, it is halved. If it is smaller, it is increased by 2.

Input HOMFLY Polynomial

$$(-3v^4(-4)+6v^4(-2))+(-3v^4(-4)+9v^4(-2)-3)z^2+(-v^4(-4)+5v^4(-2)-1)^4z^4+(v^4(-2))^4z^6$$

Fig. 10. Example of an input HOMFLY polynomial.

Identified terms:

```
[', '-3', ',', 'v^4', ',', '-4', ',', '+6', ',', 'v^4', ',', '-2', ',',
'-2', ',', '+', ',', '-3', ',', 'v^4', ',', '-4', ',', '+9', ',', 'v^4', ',',
'-2', ',', '-3', ',', ',', 'z^2', ',', '+', ',', '-v^4', ',', '-4', ',', '+5', ',', 'v^4', ',',
'-2', ',', '-1', ',', ',', 'z^4', ',', '+', ',', 'v^4', ',', '-2', ',', ',', 'z^6', ',']
```

Fig. 11. Identified terms from the input HOMFLY polynomial.

2. The R2 move adjusts the coefficients of two adjacent terms in the polynomial. If the coefficients have opposite signs, it makes them the same sign. If both are positive, it increases the smaller one. If both are negative, it decreases the larger one.
3. The R3 move applies a pattern-based adjustment to the coefficients of three consecutive terms in the polynomial. The first term's coefficient is increased by the absolute value of its v-power, the second term's coefficient is made negative, and the third term's coefficient is increased by the absolute value of its z-power.

Each transformation updates the polynomial and generates a “transform value”, calculated based on the transformation sequence. For example, an R3 move might adjust the three terms by changing their coefficients in a pre-set pattern. After each move, the modified coefficients are added together to create the transformer value, for instance, if the sum after the transformation is 43, it is converted to hexadecimal (0x2b), which is unique to each transformation sequence and will be used to alter segments of the key. This transform value is then applied to the key segment by applying the XOR function to each byte in the segment, thereby transforming it securely. Applying these moves transforms the original AES key, enhancing its security. To demonstrate the transformation process, an R3 move is applied as represented in Fig. 12. The transformed key bytes output after the application of the R3 move is represented in Fig. 13 and an example of the final transformed hexadecimal AES key is represented in Fig. 14.

To apply knot theory-based topological transformations within the cryptographic pipeline, the framework utilizes Reidemeister moves as a pre-transmission key-hardening layer. The 32-byte dynamically generated AES key is first divided into four distinct 8-byte segments, conceptually representing segments of a continuous knot loop. A HOMFLY polynomial is then introduced, where its individual terms and coefficients serve as the baseline transformation parameters.

For each 8-byte segment, a specific sequence of reversible Reidemeister moves (Type I: twisting, Type II: poking, Type III: sliding) is applied to the polynomial. Each move modifies the polynomial's coefficients according to predefined mathematical rules. Once the transformation sequence for a given segment is complete, the updated coefficients are summed and reduced modulo 256 to generate a unique transform value, denoted as τ . This transform value is subsequently applied to the corresponding key segment using a bitwise XOR operation, effectively disguising the original byte sequence.

The resulting transformed segments are concatenated to form the fully disguised key, K^* , which is then wrapped using the receiver's RSA public key for secure transmission. Because the topological moves and corresponding byte-level operations are strictly reversible, the receiver can apply the inverse operations, provided they are equipped with the correct transformation sequence and initial polynomial parameters. This process recovers the exact τ values, thereby unmasking the original 32-byte AES key for file decryption. This ensures a robust defense-in-depth layer, as an attacker must recover both the RSA wrapper and the exact topological sequence to compromise the session key.

4.5. Encryption of the transformed AES key

The transformed AES key is securely encrypted using the receiver's public key through RSA and OAEP (Optimal Asymmetric Encryption Padding) with SHA-256. This three-step encryption process – encrypting the file to be transmitted with the AES key, transforming the AES key and then encrypting the transformed key with a public key – ensures that the file remains confidential while enabling the encryption key to be transmitted securely. Both the encrypted transformed AES key and the transformation sequence used to transform the AES key are packaged together and transmitted over the established socket connection. The transformation sequence itself is encrypted alongside the transformed key using the receiver's RSA public key, ensuring that only the intended receiver, holding the corresponding RSA private key, can decrypt the package, recover the transformation sequence, and reverse the Reidemeister moves to retrieve the original AES key.

4.6. Decryption and reversal of the transformed AES key

Once the encrypted transformed AES key and transformation sequence are transmitted to the receiver, the transformed key is decrypted using the receiver's private key. Upon decryption of the transformed key, the reverse process is applied to retrieve the

```

Processing segment 1:
Segment bytes: ['0x57', '0x21', '0x70', '0x5b', '0x70', '0x7', '0x0', '0x96']

=== Applying R3 Move ===
R3 Move on segment 0:
Modified term  $-9v^0z^0$  coefficient: -9 -> -8
Modified term  $1v^0z^2$  coefficient: 1 -> -1
Modified term  $1v^0z^4$  coefficient: 1 -> 5
Byte 0: 0x57 XOR 0x2b = 0x7c
Byte 1: 0x21 XOR 0x2b = 0xa
Byte 2: 0x70 XOR 0x2b = 0x5b
Byte 3: 0x5b XOR 0x2b = 0x70
Byte 4: 0x70 XOR 0x2b = 0x5b
Byte 5: 0x7 XOR 0x2b = 0x2c
Byte 6: 0x0 XOR 0x2b = 0x2b
Byte 7: 0x96 XOR 0x2b = 0xbd

```

Fig. 12. Reidesmeister moves extracted ploynomial terms.

```

['0x7c', '0xa', '0x5b', '0x70', '0x5b', '0x2c',
'0x2b', '0xbd', '0xc', '0xd1', '0xae', '0xfa',
'0xf0', '0x6a', '0x3a', '0x5f', '0x2e', '0xe6',
'0x1d', '0xfb', '0xee', '0x31', '0x4a', '0x33',
'0x8', '0x75', '0xd4', '0xa', '0x93', '0x94', '0x48', '0x8'].

```

Fig. 13. Transformed key bytes.

```

Transformed Hex Key:
7ca5b705b2c2bbdcd1aefaf06a3a5f2ee61dfbee314a33875d4a9394488

```

Fig. 14. Example of the final transformed aes key.

```

Processing segment 1:
Segment bytes: ['0x7c', '0xa', '0x5b', '0x70', '0x5b', '0x2c', '0x2b', '0xbd']
Byte 0: 0x7c XOR 0x2b = 0x57
Byte 1: 0xa XOR 0x2b = 0x21
Byte 2: 0x5b XOR 0x2b = 0x70
Byte 3: 0x70 XOR 0x2b = 0x5b
Byte 4: 0x5b XOR 0x2b = 0x70
Byte 5: 0x2c XOR 0x2b = 0x7
Byte 6: 0x2b XOR 0x2b = 0x0
Byte 7: 0xbd XOR 0x2b = 0x96

Reversed R3 Move for segment 1:
Restored term  $-9v^0z^0$  coefficient: -8 -> -9

Reversed R3 Move for segment 1:
Restored term  $1v^0z^2$  coefficient: -1 -> 1

Reversed R3 Move for segment 1:
Restored term  $1v^0z^4$  coefficient: 5 -> 1

```

Fig. 15. Reverse transformation of the transformed key bytes of segment 1 using the R3 move.

original AES key. The transformed key is split into the same segments and the XOR transformations are reversed using the stored transform values for each segment. The Reidemeister moves are then reversed in the opposite order, restoring the polynomial to its original state for each segment. While only one Reidemeister move has been used to transform the AES key, this process can be extended by using a sequence of Reidemeister moves instead of a single move. Fig. 15 illustrates the process of reverse transforming the transformed key bytes of segment 1 using the R3 move.

The AES key retrieved from this process is used to decrypt the encrypted files, making the original content accessible to the receiver before the socket disconnects.

Table 1
Information entropy of private keys generated by different algorithms.

Method	Entropy
Chaotic System	7.9971
LCG	7.9991
Mersenne Twister	7.9955
RC4	7.9990
RSA	6.0314
DRLKeyGen (Proposed)	7.9966

Key 1
Generated Dynamic Key: 7d6a68eaf6320102db4b9a5ad5a14a7a034d9682fb2a37e524f2adfa0c1ebce7

Key 2
Generated Dynamic Key: ffb295d40e93c6c4110691d423b7e0131d773e55d906be8dcabd1b47ca85b570

Key 3
Generated Dynamic Key: 5c6cc05e823a5f53cd5bea3ac8c761851c4c570157089f246a7e4cfed41e8b09

Fig. 16. Different dynamic keys generated using the same input.

**Hamming distance between generated dynamic keys
(with same input image)**

Hamming distance between key 1 and key 2: 120
Hamming distance between key 1 and key 3: 125
Hamming distance between key 2 and key 3: 123

Average Hamming Distance: 122.66666666666667

Fig. 17. Hamming distance between keys generated using different input images and the average hamming distance between them.

5. Results and analysis

This study presents a robust and secure framework for an enhanced secure file sharing system. The structure is split into several vital fragments: dynamic key generation, file compression and extraction, file encryption and decryption, RSA key pair generation, key transformation and encryption, and file and key transmission. As the framework is a novel implementation, differentiated by its own unique combination of components, making direct comparisons with existing framework is challenging due to its complexity. However, the core aspects of this study, the dynamic key generation and key transformation, can be benchmarked against the industry standards by applying test tools to evaluate their effectiveness and security. The results and outputs for each fragment are discussed and analyzed in the following sections.

5.1. Dynamic key generation

At the fundamental level of key generation, the most crucial aspect of a security key is its randomness, ensuring that it is unpredictable and lacks patterns that could be exploited by attackers. To ensure randomness the keys are derived from a CSPRNG that produce random sequences suitable for such cryptographic application. Each key has an entropy value which determines the uncertainty of information in a system. On average it is observed that it takes about 1.90s to generate a cryptographic key and that the entropy of the generated is on par with the industry standards. Usage of deep learning algorithms further randomize the nature of the key as the interpretability of the key generation process is very low. Additionally, since security key management is only session-based, the security of the proposed workflow is less vulnerable to attacks or interceptions. Table 1 displays the information entropy of private keys generated by different algorithms.

Fig. 16 illustrates the different dynamic keys generated from the same input image. Fig. 17 exhibits the hamming distance between keys generated from different input images and the average hamming distance across them. Fig. 18 exhibits the hamming distance between keys generated from the same input image and the average hamming distance across those keys.

Table 2 displays the statistical properties of the generated CSPRN evaluated using the ent test tool. Table 3 displays the information entropy of generated random number byte array, the derived AES key and the different input images. To evaluate the strength and randomness of the CSPRNG which extends onto the randomness of the derived security key, the dieharder test

Hamming distance between generated dynamic keys	
Hamming distance between key 1 and key 2:	120
Hamming distance between key 1 and key 2:	129
Hamming distance between key 1 and key 3:	120
Hamming distance between key 1 and key 4:	120
Hamming distance between key 1 and key 5:	127
Hamming distance between key 2 and key 3:	123
Hamming distance between key 2 and key 4:	127
Hamming distance between key 2 and key 5:	130
Hamming distance between key 3 and key 4:	132
Hamming distance between key 3 and key 5:	137
Hamming distance between key 4 and key 5:	135
Average Hamming Distance: 128.0	

Fig. 18. Hamming distance between keys generated with the same input image.

Table 2
Statistical properties of the generated CSPRN evaluated using ENT test tool.

Statistic	Value
Entropy	7.99669
Chi square distribution	104857600
Arithmetic mean value	127.7405
Monte Carlo value for Pi	3.133939023
Serial Correlation coefficient	0.000070

suits tools has been employed, and the results are tabulated in Table 4 Table 5 Table 6. Based on the results obtained from the test it is observed that the CSPRNG demonstrates strong cryptographic properties. Passing all tests without anomalies suggests that the data maintains statistical randomness across many different analyses including bit distribution, sequence testing, and frequency tests. The uniformly distributed p-values between 0.01 and 0.99, suggest that there is no clustering around extreme probabilities.

To strengthen the evaluation, the proposed framework was compared with other dynamic key generation mechanisms, namely RSA-KEM, ECDHE, and static AES keying. A detailed feature-by-feature comparison covering entropy, forward secrecy, replay resistance, overhead, and compromise scope is provided in the supplementary information (Section 4). This comparison confirms that the proposed framework achieves consistently high entropy, maintains strong replay resistance, and introduces negligible overhead while preserving forward secrecy, thereby contextualizing its advantages against widely deployed methods.

5.2. File compression and extraction

With respect to file compression, the most important attributes to look out for is the compression ratio to measure the effectiveness of the various file compression algorithms when applied to different file formats. By examining the compression performance of the zlib, lzma and bz2 on formats commonly used for media, text and document files, to identify the most suitable compression technique for each file type based on its unique characteristics. The results show that text files are getting a significant performance advantage from additional compression, and the most straightforward textual formats obtain a compression ratio under 0.49% for lzma. On the other hand, those file types of which the format contains built-in and lossy compression, MP4, PNG, JPEG and MP3, respectively, show minimal size reduction, with ratios between 90%–100%, meaning almost no more compression can be made. Generally, lzma is sometimes executed slightly better than zlib and bz2, but the differences are not dramatic; hence, the algorithm choice may be somewhat flexible based on several parameters, such as speed or resource usage. The metrics of the compression are tabulated in Table 7.

5.3. File encryption

To evaluate the strength of the file encryption, various tests have been performed, namely:

(1) KNOWN-PLAINTEXT TEST This test evaluates the effectiveness of the encryption scheme to obscure relationships between plaintext and ciphertext. Two similar plaintexts were encrypted and the resulting ciphertexts were found to be distinctly different,

Table 3

Dynamically generated keys from different input images and their corresponding derived key.

Input image	Generated dynamic key	Entropy value of the generated CSPRN
	ab28bd0587cebac7e73fa7cb6aa84fe4ca1d1ec4f9ee07fd243dd1489178c0ed	7.99659
	fda9f44e6f77fa715b13943bc9ccdcdd8f39d899621a3a2c60b158da90c094	7.99624
	7c699d68fb1cf9d70383f398d8b9c1d5a50b199e52e649b546fe1d9a3e5d5959	7.99669
	656e0bfcf7cd01b57d411c12558af94d8697e8697d80a981d0dee2da33120f1c	7.99651
	41b97eb65de639cd63e7ae430a3a7617661f65ccaab729ef888cdbd7f41a8981	7.99657

indicating that the encryption algorithm incorporates sufficient randomness and complexity. This characteristic is crucial for preventing attackers from deducing patterns or relationships that could compromise the encryption's integrity.

(2) **DIFFERENTIAL ANALYSIS** This analysis measures the encryption algorithm's sensitivity to changes in plaintext. The results demonstrate a strong avalanche effect, where small alterations in the plain text caused substantial differences in the ciphertext. This behavior is an important characteristic of secure encryption algorithms as it ensures minor modification in input propagate widely throughout the ciphertext, making it virtually impossible to infer plaintext similarities from the ciphertext alone.

(3) **KEY SENSITIVITY ANALYSIS** The key sensitivity analysis assesses how changes in the encryption key affect the ciphertext. The findings revealed that even slight modifications to the encryption key result in entirely different ciphertexts. This property is critical for maintaining the strength of the encryption, as it ensures that guessing or brute-forcing a key does not produce predictable patterns. High key sensitivity also means that unauthorized access is significantly more challenging, as the encryption system is highly dependent on the precise key used.

(4) **REPLAY ATTACK TEST** The replay attack test examines whether the encryption produces identical ciphertexts for the same plaintext when encrypted multiple times with the same key. The test results confirmed that different ciphertexts were generated in each instance, indicating that the encryption algorithm employs techniques such as initialization vectors (IVs) or randomization to introduce uniqueness. This protects the encrypted data from replay attacks by ensuring that the ciphertext cannot be reused or predicted, even if the plaintext and key remain constant.

(5) **CRYPTOGRAPHIC ALGORITHM VALIDATION** The cryptographic algorithm has been validated by combining entropy based and statistical analyses to evaluate the ciphertext's properties. On average the ciphertext achieves near maximum value of 7.99878

Table 4

Dieharder test suite results of the generated cryptographically secure pseudo random number (Table: entries 1-114).

Sr. No.	Test_name	Ntup	Tsamples	Psamples	P-value	Assessment
1	diehard_birthdays	0	100	100	0.522362042	PASSED
2	diehard_operm5	0	1 000 000	100	0.4391163	PASSED
3	diehard_rank_32 × 32	0	40 000	100	0.751339026	PASSED
4	diehard_rank_6 × 8	0	100 000	100	0.456281868	PASSED
5	diehard_bitstream	0	2 097 152	100	0.689585856	PASSED
6	diehard_opso	0	2 097 152	100	0.563190846	PASSED
7	diehard_oqso	0	2 097 152	100	0.74589396	PASSED
8	diehard_dna	0	2 097 152	100	0.521261848	PASSED
9	diehard_count_1s_str	0	256 000	100	0.46198783	PASSED
10	diehard_count_1s_byt	0	256 000	100	0.556479342	PASSED
11	diehard_parking_lot	0	12 000	100	0.805015982	PASSED
12	diehard_2dsphere	2	8000	100	0.394254372	PASSED
13	diehard_3dsphere	3	4000	100	0.335745168	PASSED
14	diehard_squeeze	0	100 000	100	0.638707884	PASSED
15	diehard_sums	0	100	100	0.242700034	PASSED
16	diehard_runs	0	100 000	100	0.446693218	PASSED
17	diehard_runs	0	100 000	100	0.413300246	PASSED
18	diehard_craps	0	200 000	100	0.422501518	PASSED
19	diehard_craps	0	200 000	100	0.517104498	PASSED
20	marsaglia_tsang_gcd	0	10 000 000	100	0.70317785	PASSED
21	marsaglia_tsang_gcd	0	10 000 000	100	0.606966806	PASSED
22	sts_monobit	1	100 000	100	0.502791984	PASSED
23	sts_runs	2	100 000	100	0.574866626	PASSED
24	sts_serial	1	100 000	100	0.691228356	PASSED
25	sts_serial	2	100 000	100	0.82755013	PASSED
26	sts_serial	3	100 000	100	0.480025544	PASSED
27	sts_serial	3	100 000	100	0.455912734	PASSED
28	sts_serial	4	100 000	100	0.888683132	PASSED
29	sts_serial	4	100 000	100	0.580414718	PASSED
30	sts_serial	5	100 000	100	0.766962632	PASSED
31	sts_serial	5	100 000	100	0.729329748	PASSED
32	sts_serial	6	100 000	100	0.41732277	PASSED
33	sts_serial	6	100 000	100	0.497324884	PASSED
34	sts_serial	7	100 000	100	0.448326804	PASSED
35	sts_serial	7	100 000	100	0.483577872	PASSED
36	sts_serial	8	100 000	100	0.417198826	PASSED
37	sts_serial	8	100 000	100	0.578591424	PASSED
38	sts_serial	9	100 000	100	0.403387874	PASSED
39	sts_serial	9	100 000	100	0.347164242	PASSED
40	sts_serial	10	100 000	100	0.757677372	PASSED
41	sts_serial	10	100 000	100	0.40399579	PASSED
42	sts_serial	11	100 000	100	0.379239814	PASSED
43	sts_serial	11	100 000	100	0.330096154	PASSED
44	sts_serial	12	100 000	100	0.744362452	PASSED
45	sts_serial	12	100 000	100	0.77336536	PASSED

bits per byte over multiple test files, confirming that the data is highly random. The high entropy indicates the absence of patterns, ensuring that the original data is securely masked by the encryption. The statistical analyses via ent reveals several critical indicators of randomness and strong encryption. The ciphertext is deemed incompressible, as confirmed by the optimum compression analysis, which is a defining characteristic of secure and effective encryption. The chi-square distribution results indicate that the byte values are uniformly distributed, further validating the randomness of the ciphertext. Additionally, the arithmetic means of the data bytes, approximately 127.5, aligns with the expected value for truly random data. The Monte Carlo simulation for Pi corroborates the statistical randomness of the data by producing results consistent with expected outcomes. Lastly, the serial correlation coefficient, with a near-zero value, confirms the absence of correlation between consecutive bytes, ensuring the data remains unpredictable. Together, these metrics highlight the robustness and effectiveness of the encryption in masking patterns and maintaining randomness.

To clarify the specific contributions of the proposed framework within the broader security architecture, it is essential to specify the cryptographic properties inherited from the standard primitives versus those introduced by the dynamic key generation and topological hardening mechanisms.

The intrinsic properties of the Advanced Encryption Standard operating in Galois Counter Mode (AES-GCM) secure the data plane. Specifically, AES provides robust data confidentiality, strong diffusion through the avalanche effect, and resistance against known-plaintext and chosen-plaintext attacks. Furthermore, the GCM authentication tags ensure data integrity and immediate tamper detection, automatically rejecting any modified payloads during transmission.

In contrast, the properties achieved directly by the proposed framework secure the key plane. The Deep Reinforcement Learning based generator ensures high entropy and strict per-session key uniqueness. This prevents cross-session correlation and provides

Table 5

Dieharder test suite results of the generated cryptographically secure pseudo random number (continuation).

Sr. No.	Test_name	Ntup	Tsamples	Psamples	P-value	Assessment
46	sts_serial	13	100 000	100	0.61439989	PASSED
47	sts_serial	13	100 000	100	0.667230848	PASSED
48	sts_serial	14	100 000	100	0.463151162	PASSED
49	sts_serial	14	100 000	100	0.645476332	PASSED
50	sts_serial	15	100 000	100	0.450596556	PASSED
51	sts_serial	15	100 000	100	0.642511938	PASSED
52	sts_serial	16	100 000	100	0.757577046	PASSED
53	sts_serial	16	100 000	100	0.589320988	PASSED
54	rgb_bitdist	1	100 000	100	0.524774476	PASSED
55	rgb_bitdist	2	100 000	100	0.551406294	PASSED
56	rgb_bitdist	3	100 000	100	0.461656116	PASSED
57	rgb_bitdist	4	100 000	100	0.386423058	PASSED
58	rgb_bitdist	5	100 000	100	0.695633694	PASSED
59	rgb_bitdist	6	100 000	100	0.765791216	PASSED
60	rgb_bitdist	7	100 000	100	0.522785394	PASSED
61	rgb_bitdist	8	100 000	100	0.651704384	PASSED
62	rgb_bitdist	9	100 000	100	0.319673216	PASSED
63	rgb_bitdist	10	100 000	100	0.521763572	PASSED
64	rgb_bitdist	11	100 000	100	0.454679956	PASSED
65	rgb_bitdist	12	100 000	100	0.6152608	PASSED
66	rgb_minimum_distance	2	10 000	1000	0.486325894	PASSED
67	rgb_minimum_distance	3	10 000	1000	0.579988974	PASSED
68	rgb_minimum_distance	4	10 000	1000	0.581978308	PASSED
69	rgb_minimum_distance	5	10 000	1000	0.386603192	PASSED
70	rgb_permutations	2	100 000	100	0.493251372	PASSED
71	rgb_permutations	3	100 000	100	0.775420738	PASSED
72	rgb_permutations	4	100 000	100	0.550382258	PASSED
73	rgb_permutations	5	100 000	100	0.613853738	PASSED
74	rgb_lagged_sum	0	1 000 000	100	0.271371576	PASSED
75	rgb_lagged_sum	1	1 000 000	100	0.65714286	PASSED
76	rgb_lagged_sum	2	1 000 000	100	0.24119993	PASSED
77	rgb_lagged_sum	3	1 000 000	100	0.35208523	PASSED
78	rgb_lagged_sum	4	1 000 000	100	0.8417183	PASSED
79	rgb_lagged_sum	5	1 000 000	100	0.441337056	PASSED
80	rgb_lagged_sum	6	1 000 000	100	0.830047178	PASSED
81	rgb_lagged_sum	7	1 000 000	100	0.51943755	PASSED
82	rgb_lagged_sum	8	1 000 000	100	0.750963078	PASSED
83	rgb_lagged_sum	9	1 000 000	100	0.60501042	PASSED
84	rgb_lagged_sum	10	1 000 000	100	0.591011144	PASSED
85	rgb_lagged_sum	11	1 000 000	100	0.74035192	PASSED
86	rgb_lagged_sum	12	1 000 000	100	0.616129376	PASSED
87	rgb_lagged_sum	13	1 000 000	100	0.70590493	PASSED
88	rgb_lagged_sum	14	1 000 000	100	0.56924377	PASSED
89	rgb_lagged_sum	15	1 000 000	100	0.46182871	PASSED
90	rgb_lagged_sum	16	1 000 000	100	0.6432475	PASSED

forward secrecy, as no static key material is stored or reused across transfers. Additionally, the knot theory based topological transformation introduces an independent obfuscation layer prior to transmission. This layer ensures that even if the transport encryption wrapper is compromised, an attacker must still determine the exact Reidemeister move sequence and polynomial parameters to recover the usable AES key. Consequently, the proposed framework substantially increases resistance against key recovery, key prediction, and replay attacks, complementing the intrinsic data protection offered by AES-GCM.

5.4. Key transformation

The transformation of the key is designed to provide sufficient modification to prevent retrieval of the original key through attacks. Fig. 19 displays the hamming distance between the AES key and the transformed key, indicating that a substantial modification is observed.

5.5. Encrypted file and key transmission

The encrypted transformed AES key and files are transmitted to the receiver. To evaluate the quality of the connection between devices, network diagnostics were performed, with key metrics such as latency and packet loss recorded. The network diagnostic results between the devices are outlined in Table 8. The results indicate optimal conditions for data transmission. Figs. 20 and 21 illustrate an example of the file transfer over a secure socket connection. The server starts up, waits for a client to connect, and

Table 6

Dieharder test suite results of the generated cryptographically secure pseudo random number (continuation).

Sr. No.	Test_name	Ntup	Tsamples	Psamples	P-value	Assessment
91	rgb_lagged_sum	17	1 000 000	100	0.381393636	PASSED
92	rgb_lagged_sum	18	1 000 000	100	0.3356131	PASSED
93	rgb_lagged_sum	19	1 000 000	100	0.722456992	PASSED
94	rgb_lagged_sum	20	1 000 000	100	0.419380004	PASSED
95	rgb_lagged_sum	21	1 000 000	100	0.712895632	PASSED
96	rgb_lagged_sum	22	1 000 000	100	0.576454524	PASSED
97	rgb_lagged_sum	23	1 000 000	100	0.390945324	PASSED
98	rgb_lagged_sum	24	1 000 000	100	0.679840236	PASSED
99	rgb_lagged_sum	25	1 000 000	100	0.36206426	PASSED
100	rgb_lagged_sum	26	1 000 000	100	0.664123834	PASSED
101	rgb_lagged_sum	27	1 000 000	100	0.509407096	PASSED
102	rgb_lagged_sum	28	1 000 000	100	0.520822292	PASSED
103	rgb_lagged_sum	29	1 000 000	100	0.600689284	PASSED
104	rgb_lagged_sum	30	1 000 000	100	0.481342072	PASSED
105	rgb_lagged_sum	31	1 000 000	100	0.490217518	PASSED
106	rgb_lagged_sum	32	1 000 000	100	0.619538036	PASSED
107	rgb_kstest_test	0	10 000	1000	0.669012622	PASSED
108	dab_bytedistrib	0	51 200 000	1	0.663220034	PASSED
109	dab_dct	256	50 000	1	0.57671645	PASSED
110	dab_filltree	32	15 000 000	1	0.481629368	PASSED
111	dab_filltree	32	15 000 000	1	0.38627446	PASSED
112	dab_filltree2	0	5 000 000	1	0.36112041	PASSED
113	dab_filltree2	1	5 000 000	1	0.389346148	PASSED
114	dab_monobit2	12	65 000 000	1	0.425784006	PASSED

Table 7

Compression ratios of various file formats using ZLIB, LZMA, and BZ2 compression algorithms.

File format	File size	zlib	lzma	bz2
.mp4	9.3 MB	93.08%	90.47%	92.11%
.txt	45 KB	0.58%	0.49%	0.58%
.pdf	865 KB	71.48%	69.48%	73.38%
.png	15.6 MB	100.02%	100%	100.42%
.jpg	298 KB	99.50%	99.58%	100.05%
.jpeg	2 MB	99.49%	99.43%	98.96%
.mp3	1.45 MB	97.89%	95.62%	95.31%

Hamming distance between AES key and Transformed Key

AES Key: 495e5cdec2063536f969b878f7836858216fb4a0d90815c702d48bdc2a389ac1

Transformed Key: 677072f0ec281b18f363b272fd8962524907dcc8b1607daf5482dd8a7c6ecc97

Hamming Distance: 104.0

Fig. 19. Hamming distance between the AES key and Transformed key.

connection is established, it begins receiving a file sent by the client. As the transfer progresses, the server monitors and displays the file's reception status. When the file transfer is complete, the server saves the file and confirms that it has been received. Afterward, the client disconnects, and the server session ends, indicating a successful and orderly completion of the transfer process.

5.6. Experimental setup

To ensure the reproducibility and reliability of the performance evaluations, the hardware and software environments utilized during the experiments are defined below. The framework was developed and executed using Python 3.11.9 within a Jupyter Notebook runtime environment. The deep reinforcement learning and convolutional neural network modules were implemented using TensorFlow and Keras, while feature extraction was leveraged using OpenCV and Scikit-Image. All cryptographic operations, including AES-GCM encryption, RSA key wrapping, and PBKDF2 key derivation, were executed using the standard Python Cryptography library. The hardware environment consisted of an Intel Core i9 processor with 16 GB of system memory, and an NVIDIA RTX 3060 graphics processing unit, operating on a Windows 11 64-bit system. To ensure statistical significance and eliminate the impact of temporary system anomalies, all performance metrics reported in this study, including average key generation time, file compression ratios, and network transmission speeds, were established by averaging the results in 100–110 independent experimental trials.

```

Available commands:
1. send - Send a file
2. exit - Disconnect and exit
Sending file: ../Assets/Test-File-1.zlib.encrypted (8898103 bytes)
Progress: 8898103/8898103 bytes sent (100.0%)
Waiting for server confirmation...
File transfer successful

```

Fig. 20. Secure file transfer between devices (Sender's end).

```

Server started on 0.0.0.0:5000

Waiting for client connection...
Connected to ('127.0.0.1', 43686)

Receiving file: Test-File-1.zlib.encrypted (8898103 bytes)
Progress: 8898103/8898103 bytes received (100.0%)
Received file: Test-File-1.zlib.encrypted

Client ('127.0.0.1', 43686) disconnected
Server stopped

```

Fig. 21. Secure file transfer between devices (Receiver's end).

Table 8
Network diagnostic results between devices during transmission.

Parameter	Value
Minimum Latency	0.03 ms
Maximum Latency	0.30 ms
Average Latency	0.09 ms
Packet Loss	0.00%

Table 9
File transfer time and average speed for various file formats over the secure socket connection.

File format	File size	Time taken	Average speed
.mp4	8.9 MB	4.6 s	1.84 MB/s
.txt	0 MB	0.54 s	0.00 MB/s
.pdf	0.62 MB	0.52 s	1.19 MB/s
.png	15.34 MB	5.62 s	2.73 MB/s
.jpg	0.29 MB	0.43 s	0.67 MB/s
.jpeg	2.02 MB	1.11 s	1.82 MB/s
.mp3	1.39 MB	0.79 s	1.75 MB/s

The average time taken to transfer encrypted files of different formats and sizes are outlined in Table 9. Upon receiving the encrypted transformed AES key and the encrypted file, the receiver uses their private key to decrypt and retrieve the transformed AES key. This transformed key is then reverse-transformed to derive the original AES key. Subsequently, the original AES key is utilized to decrypt the file, making it accessible to the user in its original form. In practical terms, the use of neural networks and DRL directly strengthens the security of the framework. The CNN extracts diverse features that drive the DRL agent to generate unpredictable key material, while the entropy-based reward ensures that the produced streams avoid repeatable patterns. As a result, every session key is unique, ephemeral, and resistant to prediction even by advanced machine learning models. This was reflected in our experiments, where high entropy values and large Hamming distances confirmed the absence of cross-session correlations. Equally important, the adaptive nature of DRL allows the system to maintain target randomness under different operating conditions, including when inputs vary or anomalies are introduced. In practice, this means that keys remain difficult to infer even if traffic is partially leaked, and that the system can refresh keys quickly during reconnection or replay events. Together, these properties demonstrate how the learning-based approach translates into real-world protections against brute-force guessing, replay attempts, and key leakage, while keeping key generation efficient enough for real-time file sharing. The adaptability of the proposed framework is reflected in its operational behavior under conditions that adversaries would typically exploit in real time. Each session employs a fresh DRL-driven key stream that is mixed with PBKDF2 to derive a 256-bit AES key and salt, ensuring that no material is stored or reused. If a session is interrupted or reconnected, a new RSA pair and AES key are generated, thereby reducing the attacker's window. Online tampering is countered by AES-GCM authentication, which immediately rejects modified payloads and triggers a rekey; the probability of a successful forgery remains negligible when proper nonce use is enforced. Replay attempts are defeated by unique per-session nonces,

while cross-session correlation is prevented by the DRL generator, which is explicitly rewarded for entropy and low predictability. Empirical evaluation confirms these properties: entropy values consistently approach 7.996–7.999 bits per byte, Dieharder tests return uniformly distributed p-values, and large Hamming distances across sessions indicate low correlation. Execution time for each session key averages 1.90 s, which does not affect throughput since the data plane remains standard AES-GCM. Replay and tamper trials further demonstrate distinct ciphertexts and immediate authentication failure on modification. Collectively, these outcomes confirm that the system sustains robustness against real-time adversarial disruptions while preserving the performance characteristics of conventional symmetric encryption.

6. Conclusion and future scope

In conclusion, this study introduces an innovative framework that integrates deep learning and topological transformations using knot theory to address critical limitations in traditional cryptographic methods. Utilizing these methods overcomes the vulnerabilities with static keys and provides an enhanced and adaptive solution for secure device to device file-sharing. Conventional approaches that use techniques like AES and RSA depend on static keys, which make the system susceptible to interception over time. The proposed framework mitigates these risks by employing a session-based system that dynamically generates unique security keys, ensuring that each transmission is individually protected, thus significantly reducing the potential for unauthorized access. Moreover, one vital advantage of this framework is that the files are encrypted on the device before transmission. This mode of device-side encryption ensures that the files are never exposed, as they never leave the device in an unencrypted state. Furthermore, the AES key is transformed using knot theory, increasing its complexity such that, even if the attacker were to decrypt the RSA layer, the mathematical layer needs to be re-transformed using the exact polynomial and transformation sequence to retrieve the original AES key back, making it highly secure against interceptions. While the present evaluation focused on secure file sharing, the framework was described as being adaptable to mobile and distributed domains. Based on the design characteristics: the session-based key generation produces lightweight random streams, the knot transformation operates in constant time on only 32 bytes per session, and the socket-driven architecture treats each sender–receiver pair as an independent session. These factors suggest that the framework can be integrated into resource-constrained or distributed environments with minimal modification, although full validation in such settings remains as a direction for future work.

Future efforts could enhance this framework by incorporating more cutting edge deep learning models, such as transformers or GANs, to improve security key diversity and randomness and expand the principles of topological transformations. Furthermore, this can be extended by developing a server-based file sharing system that employs the proposed framework to help transfer files over longer ranges. Additionally, its application could be widened by integrating it with cloud storage, IoT networks, and distributed ledger systems. Specifically, exploring the integration of dynamic key generation and topological hardening within the Internet of Medical Things (IoMT) and healthcare environments represents a highly promising avenue for future research. Recent advancements have demonstrated the efficacy of combining deep learning, deep reinforcement learning, and blockchain for multi-layered security and intrusion detection in these domains [32–34]. Since these specialized architectures have not yet been fully explored within the current scope, adapting the proposed framework to complement these decentralized healthcare systems will be a primary focus of subsequent investigations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors gratefully acknowledge Sri Mata Amritanandamayi Devi (Amma), Chancellor, Amrita Vishwa Vidyapeetham, for her inspiration and for providing financial support for the Article Processing Charges (APC) of this publication.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.rico.2026.100744>.

Data availability

Data will be made available on request.

References

- [1] HB BG, Poornachandran P, KP S. Deep-net: deep neural network for cyber security use cases. 2018, arXiv preprint arXiv:1812.03519.
- [2] Spasova G, Karova M. A new secure image encryption model based on symmetric key. In: 2021 International Conference on Biomedical Innovations and Applications. 2022, p. 107–10.
- [3] Katz J, Lindell Y. Introduction to modern cryptography. second ed. CRC Press; 2015.
- [4] Daemen J, Rijmen V. The design of rijndael, AES - the advanced encryption standard. Springer-Verlag; 2002.
- [5] Prasad SD Vara, Kavitha J, Yarava RK, Nagaraju A, Babu GC, Krishna P Gopala. A secured method for authentication of data security using public key cryptography. In: 2023 Third International Conference on Artificial Intelligence and Smart Energy. 2023, p. 1413–20. <http://dx.doi.org/10.1109/ICAIS56108.2023.10073781>.
- [6] Jayanthi S, Nageswarvijay S, Kumar RKR, Kanth RK. Smart key using AES algorithm. In: 2015 Third International Conference on Inventive Research in Computing Applications. 2021, p. 467–73.
- [7] Vivek Sridhar Mallya PV, Ajith A, Sangeetha TR, Krishnan A, Narayanan G. Implementation of differential privacy using Diffie-Hellman and AES algorithm. In: Inventive communication and computational technologies: proceedings of ICICCT 2019. Singapore: Springer; 2020, p. 143–52.
- [8] Xue M, Zhu C. The socket programming and software design for communication based on client/server. In: 2009 Pacific-Asia Conference on Circuits, Communications and Systems. 2009, p. 775–7. <http://dx.doi.org/10.1109/PACCS.2009.89>.
- [9] Yao AC-C, Zhao Y. Privacy-preserving authenticated keyexchange over internet. IEEE Trans Inf Forensics Secur 2014;9(1):125–40. <http://dx.doi.org/10.1109/TIFS.2013.2293457>.
- [10] Naranjo JAM, López-Ramos JA, Casado LG. Applications of the extended euclidean algorithm to privacy and secure communications. In: Proc. of 10th international conference on computational and mathematical methods in science and engineering. 2010, p. 702–13.
- [11] Ding Y, Tan F, Qin Z, Cao M, Choo KKR, Qin Z. DeepKeyGen: a deep learning-based stream cipher generator for medical image encryption and decryption. IEEE Trans Neural Netw Learn Syst 2021;33(9):4915–29.
- [12] Chunka C, Goswami RS, Banerjee S. An efficient mechanism to generate dynamic keys based on genetic algorithm. Secur Priv 2021;4(5):e37.
- [13] Almaraz Luengo E, Román Villalán J. Cryptographically secured pseudo-random number generators: Analysis and testing with NIST statistical test suite. Mathematics 2023;11(23):4812. <http://dx.doi.org/10.3390/math11234812>.
- [14] Yu H, Kim Y. New RSA encryption mechanism using one-time encryption keys and unpredictable bio-signal for wireless communication devices. Electronics 2020;9(2):246.
- [15] Zhou S, Wang X, Zhang Y, Ge B, Wang M, Gao S. A novel image encryption cryptosystem based on true random numbers and chaotic systems. Multimedia Syst 2022;1–18.
- [16] Ullah F, Pun CM. Deep self-learning based dynamic secret key generation for novel secure and efficient hashing algorithm. Inform Sci 2023;629:488–501.
- [17] Karabey I, Akman G. A cryptographic approach for secure client-server chat application using public key infrastructure (pki). In: 2016 11th international conference for internet technology and secured transactions. IEEE; 2016, p. 442–6.
- [18] Wu X, Han Y, Zhang M, et al. Pseudorandom number generators based on neural networks: a review. J King Saud Univ Comput Inf Sci 2025;37:18. <http://dx.doi.org/10.1007/s44443-025-00007-4>.
- [19] Bura RO, Nida HS. Image transmission using base64 encoding and advanced encryption standard algorithm based on socket programming. In: 2021 6th International Workshop on Big Data and Information Security. 2021, p. 115–20. <http://dx.doi.org/10.1109/IWBIS53353.2021.9631846>.
- [20] Xiao Qingjiang, Zhao Jinrong, Feng Sheng, Li Guyue, Hu Aiqun. Securing NextG networks with physical-layer key generation: A survey. Secur Saf 2024;3:2023021. <http://dx.doi.org/10.1051/sands/2023021>.
- [21] Bhardwaj A, Som S. Study of different cryptographic technique and challenges in future. In: 2016 International Conference on Innovation and Challenges in Cyber Security. 2016, p. 208–12. <http://dx.doi.org/10.1109/ICICCS.2016.7542353>.
- [22] Sconza S, Wildi A. Knot-based key exchange protocol. 2024, Retrieved from <https://eprint.iacr.org/2024/471>.
- [23] Shaheen SH, Yousaf M, Majeed MY. Comparative analysis of internet key exchange protocols. In: 2015 International Conference on Information and Communication Technologies. 2015, p. 1–6. <http://dx.doi.org/10.1109/ICICT.2015.7469595>.
- [24] Saeed M, Shahhoseini HS, Mackvandi A, Rezaeinezhad MR, Naddafium M, Bidoki MZ. A secure two-party password authenticated key exchange protocol. In: Proceedings of the 2014 IEEE 15th international conference on information reuse and integration. 2014, p. 466–74. <http://dx.doi.org/10.1109/IRI.2014.7051926>.
- [25] Panda M. Performance analysis of encryption algorithms for security. In: 2016 International Conference on Signal Processing Communication PoweIr and Embedded System. 2016, p. 278–84.
- [26] Ramesh A, Suruliandi A. Performance analysis of encryption algorithms for information security. In: 2013 International Conference on Circuits Power and Computing Technologies. 2013, p. 840–4.
- [27] SreeDeepika M, Lekhya NS, Likhesh K, Rajagopal SM. Implementing internet key exchange with image-based key exchange through socket programming. In: 2024 IEEE 9th international conference for convergence in technology. IEEE; 2024, p. 1–6.
- [28] Nguyen TT, Reddi VJ. Deep reinforcement learning for cyber security. IEEE Trans Neural Netw Learn Syst 2023;34(8):3779–95. <http://dx.doi.org/10.1109/TNNLS.2021.3121870>.
- [29] Oh SH, Kim J, Nah JH, Park J. Employing deep reinforcement learning to cyber-attack simulation for enhancing cybersecurity. Electronics 2024;13(3):555.
- [30] Aparna VS, Rajan A, Jairaj I, Nandita B, Madhusoodanan P, Remya AA. Implementation of AES algorithm on text and image using MATLAB. In: 2019 3rd international conference on trends in electronics and informatics. IEEE; 2019, p. 1279–83.
- [31] Weaver AC. Secure sockets layer. In: Computer, In: Computer, 2006;vol. 39, no. 4:88–90.<http://dx.doi.org/10.1109/MC.2006.138>,
- [32] Sharma Nikhil, Shambharkar Prashant Giridhar. Enhancing internet of medical things security: A multi-layered approach using dynamic adaptive deep reinforcement learning and blockchain. Comput Electr Eng 2026;129:110808. <http://dx.doi.org/10.1016/j.compeleceng.2025.110808>, Part A.
- [33] Sharma N, Shambharkar PG. Multi-attention DeepCRNN: an efficient and explainable intrusion detection framework for internet of medical things environments. Knowl Inf Syst 2025;67:5783–849. <http://dx.doi.org/10.1007/s10115-025-02402-9>.
- [34] Sharma Nikhil, Shambharkar Prashant Giridhar. Towards secure healthcare: SA-GBO-ODBN model utilizing blockchain and deep learning for data handling and diagnosis. Comput J 2025;68(10):1386–423. <http://dx.doi.org/10.1093/comjnl/bxaf045>.