

AN ALGORITHMIC FRAMEWORK FOR TEXT ENCRYPTION BASED ON A CLOSED KNIGHT'S TOUR ON OPTIMAL-SIZED BOARDS

Rawin Youngnoi*

Received: July 11, 2025; Revised: January 16, 2026; Accepted: February 09, 2026

Abstract

The application of knight's tour permutations to data encryption has been explored for media such as images and video, where algorithms typically operate on standardized, fixed-size blocks of data. This paper addresses a different and more fundamental challenge: the encryption of variable-length text, where no predefined block size exists. The primary contribution of this work is a novel algorithmic framework that determines the optimally sized rectangular board that admits a closed knight's tour and minimizes the padding required for a given text length. Using this framework, we develop a complete symmetric-key cryptosystem that combines a Vigenère-style XOR substitution for confusion with a knight's tour-based permutation for diffusion. To improve robustness, we incorporate an iterative board-resizing mechanism to handle cases in which the tour-finding heuristic fails. The system's performance reveals an efficient board-finding process that both theoretically and empirically minimizes padding, along with a non-linear increase in encryption time for input length increases. Security analysis demonstrates that the cipher obfuscates plaintext statistics and has a large keypace; however, it exhibits a poor avalanche effect, consistent with its single-round architecture. This work lies at the intersection of graph theory, optimization, and cryptography, and it concludes by proposing a multi-round architecture with improved diffusion for future research.

Keywords: Knight's Tour; Cryptography; Symmetric Key; Algorithm Design; Optimization; Graph Theory

Introduction

In the digital age, secure data transmission is critical. While modern cryptography relies heavily on principles from discrete mathematics and number theory, there is a longstanding and growing interest in creating novel encryption schemes by leveraging the complexity of combinatorial optimization and graph theory. The inherent complexity of problems such as, the Hamiltonian

path problem (Pohl, 1967), presents a rich field for developing cryptographic primitives. Graph-based encryption, where the structure of a graph itself is used for encoding, offers unique properties for ensuring data confidentiality and integrity, as explored in prior work on graph-based symmetric-key cryptography.

Department of Mathematics and Statistics, Faculty of Science and Technology, Thammasat University, Rangsit Campus, Pathum Thani 12120, Thailand, E-mail: yrawin@tu.ac.th, ORCID: 0009-0004-2851-8490

** Corresponding author*

DOI: <https://doi.org/10.55766/sujst10739>

Suranaree J. Sci. Technol. 33(4):030401(1-12)

The knight's tour problem, a special case of the Hamiltonian path problem on a chessboard graph, seeks a path for a knight on a chessboard to visit every square exactly once. This path constitutes a permutation of the board's squares, a property that has been adapted for cryptographic purposes. This concept of using a knight's tour as a permutation tool has been successfully applied to scramble data in various domains. The most common application is in image encryption, where a common approach is to divide an image into smaller, uniform blocks (e.g., 5×5 or 8×8 pixels) and apply a pre-computed knight's tour permutation to shuffle the pixels within each block (Delei *et al.*, 2008; Philip, 2013; Singh *et al.*, 2015; Oraby & Hamza, 2023). While this method is functional, it can lead to significant inefficiency, as substantial padding must be added if the image dimensions are not a perfect multiple of the block size (Singh *et al.*, 2015), thereby increasing data overhead. The versatility of the knight's tour in security applications is further demonstrated in areas such as image and video steganography, where the tour path is used to hide data within a cover image (Nie *et al.*, 2019; Younus & Younus, 2019), as well as in applications like general-purpose data scramblers (Romanuke *et al.*, 2024), dynamic S-box generation (Khan *et al.*, 2022), and encryption key generation (Mahmood & Rahim, 2017).

Finding a knight's tour is an instance of the Hamiltonian path problem, which is NP-hard in the general case (Pohl, 1967). However, for a standard chessboard, solutions can be found efficiently (Yates *et al.*, 2025). Several methods exist, ranging from simple heuristics to complex, complete algorithms. Warnsdorff's rule, a well-known heuristic, suggests moving the knight to the square with the fewest onward moves (Squirrel & Çull, 1996; Gerlach & Cape Coral, 2015). The rule is computationally efficient and often finds a tour quickly in linear time. However, as a heuristic, it is not guaranteed to find a solution for all starting positions and can fail on certain board configurations.

In contrast, complete algorithms are guaranteed to find a tour if one exists. These include methods based on backtracking or divide-and-conquer (Gordon & Slocum, 2004; Lin & Wei, 2005). Divide-and-conquer algorithms, for instance, can solve the problem on $n \times n$ in linear time ($O(n^2)$) by breaking the board into smaller solvable pieces and patching the sub-tours together, as demonstrated by Parberry (1997).

While the application of knight's tours is established, applying fixed-size permutations to variable-sized data highlights a clear research gap related to efficiency. This challenge is particularly

acute when encrypting unstructured data like plain text, for which no natural block structure exists. This raises the central question: what is the most efficient board size to use for a given message? A board that is too large creates unnecessary computational overhead and excessive padding, while one that is too small cannot contain the data. The core challenge, therefore, shifts from simply permuting the data to first solving the optimization problem of minimizing this wasted space, a theme related to other optimization studies on knight's tours (Demaio & Hippchen, 2009).

This research addresses this specific gap. Instead of using a fixed-size board, our work introduces a novel algorithmic framework that solves the optimization problem of finding the ideal grid for any given data length, based on the mathematical foundations for tours on rectangular boards (Chia & Ong, 2005). The main contributions of this paper are threefold. First, it introduces an efficient heuristic algorithm to determine the optimal $m \times n$ board size that minimizes padding for a given text length. This is in accordance with Schwenk's Theorem (Schwenk, 1991) on the existence of a closed knight's tour. Second, building upon this foundation, it details the design and implementation of a robust, complete symmetric-key cryptosystem that intelligently handles the limitations of its underlying tour-finding heuristic. Finally, the work is validated by a comprehensive performance and security analysis of the proposed system. This includes runtime benchmarks, empirical validation of the proposed algorithm's padding efficiency, and a security evaluation covering frequency analysis and the avalanche effect.

The remainder of this paper is organized as follows. The Methodology section details the theoretical analysis and core algorithms for board optimization and tour generation. This is followed by a description of the three-stage symmetric-key encryption process. Next, the Results and Discussion section presents a comprehensive evaluation of the system. This section begins with an illustrative example. It then presents quantitative analyses of the algorithm's padding efficiency, runtime performance, and security characteristics, and concludes by interpreting the findings and discussing design trade-offs. Finally, the Conclusion summarizes the work and proposes specific directions for future research.

Methodology

The proposed methodology is presented in three parts. First, we detail the system's two core

algorithmic components: the board optimization algorithm (with its formal analysis) and the tour generation algorithm. Next, we explain how these algorithms are utilized within the complete three-stage symmetric-key encryption process.

The Board Optimization Algorithm

The central challenge in encrypting variable-length text with a board-based permutation is determining the ideal (i.e., minimum-size) board. To address this, we developed an algorithm to find the smallest rectangular chessboard of m rows and n columns ($m \leq n$) that can contain a given text of length L , while satisfying the necessary and sufficient conditions for admitting a closed knight's tour.

The existence of such tours on the rectangular chessboard is characterized by Schwenk's Theorem, which provides the mathematical foundation for our constraints.

Theorem (Schwenk, 1991). An $m \times n$ chessboard with $m \leq n$ admits a closed knight's tour if and only if:

1. m and n are not both odd;
2. $m \notin \{1,2,4\}$;
3. $m \neq 3$ or $n \notin \{4,6,8\}$.

Based on Schwenk's theorem, we formulate the task as a constrained optimization problem: Given a text length L , find positive integers $m, n \in \mathbb{Z}^+$ that solve:

$$\min_{m,n \in \mathbb{Z}^+} (mn - L)$$

Subject to the following constraints:

1. $mn \geq L$.
2. $m \leq n$.
3. m or n are even.
4. $m \notin \{1,2,4\}$.
5. If $m = 3$, then $n \notin \{4,6,8\}$.

The five constraints are derived directly from the necessary and sufficient conditions established by Schwenk (1991) to ensure both the validity of the encryption mapping and the existence of a closed tour. A detailed justification for each constraint follows:

- Constraint 1 ($mn \geq L$): This is the fundamental capacity constraint. It guarantees that the number of squares on the board is sufficient to map every character of the input plaintext injectively, where $mn - L$ represents the required padding.
- Constraint 2 ($m \leq n$): This constraint imposes an ordering to exploit the geometric symmetry of the chessboard. Since the existence of a tour on an $m \times n$ board implies existence on an

$n \times m$ board, imposing $m \leq n$ reduces the search space without excluding any valid topological solutions.

- Constraint 3 (m or n are even): This satisfies the parity condition required for a closed tour (Schwenk, 1991). A chessboard graph is bipartite; a closed knight's tour constitutes a Hamiltonian cycle, which can only exist on a bipartite graph if the partitions are of equal size. This requires the total number of squares (mn) to be even, meaning m and n cannot both be odd.
- Constraint 4 ($m \notin \{1,2,4\}$): This filters out dimensions where tours are impossible due to insufficient connectivity (Schwenk, 1991). For $m = 1$ and $m = 2$, the degree of the vertices is insufficient to form a cycle. For $m = 4$, Schwenk proved that no closed tour exists regardless of n .
- Constraint 5 (If $m = 3$, then $n \notin \{4,6,8\}$): This addresses the specific forbidden subcases identified by Schwenk (1991) for boards with a narrow dimension of 3, where the knight's movement is too restricted to form a closed loop.

The optimization problem posed above is non-trivial because the feasible region defined by Schwenk's Theorem is discrete and non-convex. Unlike standard area minimization where one might simply choose the smallest integer area $A \geq L$, the constraints on m and n create gaps in the set of valid board sizes. For instance, a text of length L might numerically fit into a board of area A , but if that area requires dimensions that violate the parity or connectivity constraints (e.g., a $4 \times k$ board), that area is invalid. Therefore, the problem cannot be solved by simple algebraic inversion; it requires searching the discrete space of valid integer pairs (m, n) to find the configuration that minimizes the objective function $mn - L$ (the padding) while strictly adhering to the topological requirements of the closed knight's tour.

To solve this optimization problem, a search method is used. The derivation is based on a case-by-case analysis of the number of rows, m . For even values of m where $m \geq 6$, Constraint 1 leads to the inequality $n \geq L/m$. Since we want to minimize the size of the board, we choose the smallest integer that is greater than or equal to L/m . That is $n = \lceil L/m \rceil$. In contrast, for odd values of m where $m \geq 5$, Constraint 3 dictates that n must be an even number. Therefore, we can express n in the form $n = 2k$ for some integer k . By Constraints 1 and 2, we obtain the inequality $k \geq \max\{m/2, L/(2m)\}$. Therefore, $n = 2\lceil \max\{m/2, L/(2m)\} \rceil$. That is $n = 2\lceil L/$

$(2m)]$ if $L \geq m^2$, otherwise $n = 2\lceil m/2 \rceil = m + 1$. A special case exists for $m = 3$, which is similar to the previous case, we have that $k \geq 5$ by Constraints 3 and 5. As a result, $n = 2\lceil \max\{5, L/6\} \rceil$. Precisely, $n = 2\lceil L/6 \rceil$ for $L > 30$ or $n = 10$ for $L \leq 30$.

To solve this optimization problem efficiently, we employ a search method that iterates through possible row dimensions m . The search range is strictly bounded by the constraints derived above. According to Constraint 4, the row dimension m cannot be 1, 2, or 4. Consequently, the smallest valid value is $m = 3$. We therefore initialize the best-found solution with the $m = 3$ case before iterating through the remaining search space starting from $m = 5$. Furthermore, the iteration for m is strictly bounded by $\lceil \sqrt{L} \rceil$. This upper bound is derived from Constraint 2 ($m \leq n$). For any board with area $mn \geq L$, the smaller dimension m must satisfy $m^2 \leq mn$. Since we seek a board where the area mn is as close to L as possible, the optimal m will not significantly exceed $\sqrt{L}$. The ceiling function is applied to ensure that if the optimal board is a square (where $m = n$), the integer dimension capable of containing L is included in the search. This logic forms the basis of our efficient search algorithm, as follows.

Algorithm 1: Optimal Board Search

Input: Text length L
Output: Optimal dimensions m, n
Initialize $m \leftarrow 3$
Initialize $n \leftarrow 2\lceil \max\{5, L/6\} \rceil$
Initialize $nDummies \leftarrow mn - L$
for $i \leftarrow 5$ to $\lceil \sqrt{L} \rceil$ do
 if i is even then
 $j \leftarrow \lfloor L/i \rfloor$
 else
 $j = 2\lceil \max\{i/2, L/(2i)\} \rceil$
 end if
 if $ij - L \leq nDummies$ and $i \leq j$ then
 $m \leftarrow i$
 $n \leftarrow j$
 $nDummies \leftarrow ij - L$
 end if
end for
return m, n

This algorithm, with a time complexity of $O(\sqrt{L})$, efficiently finds a suitable board for the subsequent encryption process. Moreover, there might be several optimal boards for a given text length. The primary objective of the algorithm is to minimize the number of dummy characters ($mn - L$). However, since the algorithm iterates m from 3

upwards to $\lceil \sqrt{L} \rceil$ and updates the optimal solution whenever a new pair (m, n) is found with equal or fewer dummy characters, it inherently favors larger values of m . Since $m \leq n$, maximizing m minimizes the difference $n - m$. To be precise, if (m_1, n_1) and (m_2, n_2) are two board sizes offering the same minimal padding (and thus equal area $A = m_1n_1 = m_2n_2$) such that $m_1 < m_2$. Since $A = m_1n_1 = m_2n_2$ and $m_1 < m_2$, it follows that $n_2 < n_1$. Consequently, the difference between dimensions strictly decreases, i.e., $n_2 - m_2 < n_1 - m_1$. Therefore, in cases where multiple board sizes offer the same minimal padding, the algorithm implicitly selects the configuration where m and n are closest in value, resulting in the most square-like board possible for that efficiency level.

Analysis of Dummy Characters

The efficiency of Algorithm 1 in minimizing wasted space can be formally analyzed. The proof of the main theorem relies on understanding the padding efficiency for each potential board shape that Algorithm 1 considers. The following lemma establishes these foundational bounds for each case based on the row dimension m .

Lemma 1. Let L be the text length and let m be a potential number of rows considered by Algorithm 1. The number of dummy characters, $mn - L$, for that specific m is bounded as follows:

1. If $m \geq 6$ is an even number, then $0 \leq mn - L < m$.
2. If $m = 3$:
 - a. and $L \leq 30$, then $0 \leq mn - L < 30$.
 - b. and $L > 30$, then $0 \leq mn - L < 6$.
3. If $m \geq 5$ is an odd number:
 - a. and $L \geq m^2$, then $0 \leq mn - L < 2m$.
 - b. and $L < m^2$, then $m \leq mn - L < 3m - 1$.

Proof of Lemma 1.

Case 1: Let $m \geq 6$ be an even number. Thus, we get $n = \lceil L/m \rceil$. From the properties of the ceiling function, we have $n - 1 < L/m \leq n$, which implies $0 \leq mn - L < m$.

Case 2: Let $m = 3$. For $L \leq 30$, we have $L/(2m) \leq 5$. Thus, we get $n = 10$, and therefore, $0 \leq mn - L < 30$. For $L > 30$, we have $L/(2m) > 5$, so we have that $n = 2\lceil L/(2m) \rceil$. From the properties of the ceiling function, we have $n/2 - 1 < L/(2m) \leq n/2$, which implies $0 \leq mn - L < 6$.

Case 3: Let $m \geq 5$ be an odd number. For $L \geq m^2$, we have $n = 2\lceil L/(2m) \rceil$. From the properties of the ceiling function, we have $n/2 - 1 < L/(2m) \leq n/2$, which implies $0 \leq mn - L < 2m$.

For $L < m^2$, we have $n = m + 1$ and $mn = m^2 + m$. By Algorithm 1, m is searched from 3 to $\lceil \sqrt{L} \rceil$ excluding $m = 4$. Therefore, $m \leq \lceil \sqrt{L} \rceil < \sqrt{L} + 1$. This implies that $(m - 1)^2 < L$. Since $L < m^2$ and $mn = m^2 + m$, we have $m \leq mn - L < 3m - 1$.

While Lemma 1 establishes the bounds for each potential choice of m , the final efficiency of the system is determined by the output of the complete optimization process in Algorithm 1. The following theorem formally states the bounds on the number of dummy characters for the final selected board.

Theorem 1. Let L be the text length and let m and n be the dimensions of the final chessboard selected by Algorithm 1 after it has completed its optimization search. The number of dummy characters, $mn - L$, is bounded as follows:

1. If $L \leq 30$, then $0 \leq mn - L < 30$.
2. If $L > 30$, then $0 \leq mn - L < 6$.

Proof of Theorem 1.

The proof relies on the fact that Algorithm 1 searches a finite set of candidate dimensions that includes the baseline case of $m = 3$. Let S be the set of all valid board configurations (m, n) considered by the algorithm. The algorithm calculates the padding $P(m, n) = mn - L$ for each $(m, n) \in S$ and returns the configuration (m^*, n^*) such that $P(m^*, n^*) = \min\{P(m, n) | (m, n) \in S\}$.

Case 1: $L \leq 30$. For short texts, Schwenk's constraints heavily restrict valid boards. Constraint 4 excludes $m \in \{1, 2, 4\}$. Constraint 5 excludes $n \in \{4, 6, 8\}$ when $m = 3$. The smallest valid board satisfying all conditions is 5×6 (Area = 30). If $0 < L \leq 30$, the algorithm will minimally select this 5×6 board. Thus, $0 \leq mn - L < 30$.

Case 2: $L > 30$. We prove this case by contradiction. Let (m^*, n^*) be the optimal board dimensions returned by Algorithm 1, and let D^* be the resulting number of dummy characters, i.e., $D^* = P(m^*, n^*)$. Assume that $D^* \geq 6$. Algorithm 1 explicitly initializes the search space with $m = 3$. Let D_3 denote the number of dummy characters corresponding to the optimal board where $m = 3$. According to Lemma 1, for $m = 3$ and $L > 30$, there exists a valid dimension n such that $0 \leq D_3 < 6$. Since (m^*, n^*) is the output from Algorithm 1, this means that $D^* \leq D_3$. Combining these inequalities yields $D^* < 6$. This contradicts the initial assumption that $D^* \geq 6$. Thus, it must be that for any $L > 30$, the number of dummy characters is fewer than 6.

The results from Theorem 1 provide a comprehensive picture of the padding efficiency of Algorithm 1. The key insight is that for any

message longer than 30 characters, the algorithm is highly efficient, guaranteeing the number of wasted cells (dummy characters) is very small (fewer than 6). The only scenario where a larger number of dummy characters may occur is for very short texts ($L \leq 30$), where the smallest possible valid board (5×6) must be used.

The Tour Generation Algorithm

While Schwenk's Theorem guarantees that a closed tour exists for the selected board dimensions, a simple heuristic like Warnsdorff's rule (Squirrel & Çull, 1996) is not guaranteed to find it from every starting point. Therefore, a robust process involving a systematic search over all possible starting points is employed to ensure a tour is always generated.

The core of this search utilizes Warnsdorff's rule, which moves the knight to the square with the fewest onward moves, as shown in Figure 1. To ensure the process is deterministic, a tie-breaking rule is applied if multiple moves have the same minimum degree: the algorithm selects the move with the smallest row index, and then the smallest column index if a tie still exists.

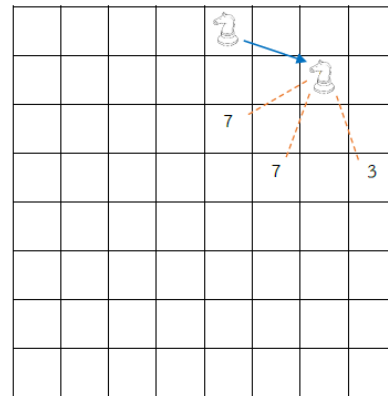


Figure 1. Example of a knight's move using Warnsdorff's rule

The algorithm iterates through every square on the board, using each as a potential starting point for the heuristic. If a valid closed tour is found, the search succeeds and this tour path is used. However, in the rare case that the search exhausts all starting points without finding a tour, the algorithm initiates a board-resizing loop. It finds a new board using Algorithm 1 with a new target length of $L = mn + 1$ and repeats the tour search process. This robust, iterative approach guarantees a successful outcome. The complete logic is formally detailed in Algorithm 2.

Algorithm 2: Closed Knight's Tour Search**Input:** Board Dimensions (m, n) **Output:** Tour path (an ordered list of coordinates) or an empty list on failure

Potential_Starts $\leftarrow$ an ordered list of all $m \times n$ squares, from $(0,0)$ to $(m-1, n-1)$
 for each *start_square* in *Potential_Starts* do
 path $\leftarrow$ an empty list of coordinates
 visited_board $\leftarrow$ an $m \times n$ matrix of FALSE values
 current_square $\leftarrow$ *start_square*
 Add *current_square* to *path*
 Set *visited_board* at *current_square* to TRUE

 for $i \leftarrow 2$ to mn do
 next_moves $\leftarrow$ set of valid, unvisited knight moves from *current_square*

 if *next_moves* is empty then
 break
 end if
 min_degree $\leftarrow \infty$
 best_move $\leftarrow$ NULL
 for each *move* in *next_moves* do
 onward_moves $\leftarrow$ set of valid, unvisited moves from *move*
 d $\leftarrow$ |*onward_moves*|
 if $d < \text{min_degree}$ then
 min_degree $\leftarrow d$
 best_move $\leftarrow$ *move*
 else if $d == \text{min_degree}$ then
 best_move $\leftarrow$ tie break between *best_move* and *move*
 end if
 end for
 current_square $\leftarrow$ *best_move*
 Add *current_square* to *path*
 Set *visited_board* at *current_square* to TRUE
 end for
 if |*path*| == $m \times n$ and *start_square* is reachable from *current_square* then
 return *path*
 end if
 end for
 return empty array

The Symmetric-Key Encryption Process

The encryption process is a three-stage symmetric-key cipher that operates in the sequence: Pad $\rightarrow$ Substitute $\rightarrow$ Permute. It utilizes two core algorithms detailed previously: the Board Optimization Algorithm (Algorithm 1) and the Tour Generation Algorithm (Algorithm 2).

The system uses two secret keys provided by the user to control the encryption process. The first is the Substitution Key (s), which is a password that serves as the key for the Vigenère-style XOR encryption. The second is the Shift Key (r), an integer that defines the rotational distance for the final character permutation along the knight's tour path.

The encryption process begins with padding. A single, non-printable *NULL* character is appended to the end of the original text message. This character acts as an unambiguous end-of-message delimiter. The length of this new text, L_{new} , is then used as the initial input for the board finalization process.

Following this, the system enters a robust, iterative loop to determine the final board size and tour path. First, the Board Optimization Algorithm is executed to find an optimal $m \times n$ board for the current target length. Then, the Tour Generation Algorithm is executed on this $m \times n$ board to find a valid closed knight's tour. In the rare case that the tour-finding heuristic fails for that specific board configuration, the system handles this failure by updating the target length to $L_{current} = mn + 1$ and repeating this loop. This process continues until a suitable board and its corresponding closed tour are successfully generated, ensuring a reliable outcome. Once the final $m \times n$ board is determined, we place the message on the board, one character per square, from top to bottom, left to right. The remaining text block is filled to its full $m \times n$ size with dummy characters. These dummy characters are chosen randomly from the set of characters already present in the original message to increase security against statistical analysis.

The next stage is substitution. The entire $m \times n$ data block, which now contains the message, the delimiter, and random padding, is encrypted using a Vigenère-style XOR cipher with the Substitution Key (s). This polyalphabetic substitution provides confusion by ensuring that identical characters in the plaintext are mapped to different characters in the ciphertext. The key s is repeated as necessary to match the length of the data block.

For example, to encrypt the plaintext "TEXT" with the key "KEY", the operation is performed on their numerical byte values:

- Plaintext "T" (ASCII 84) XOR Key "K" (ASCII 75) = 21.
- Plaintext "E" (ASCII 69) XOR Key "E" (ASCII 69) = 0 (a *NULL* character).
- Plaintext "X" (ASCII 88) XOR Key "Y" (ASCII 89) = 1.

- Plaintext "T" (ASCII 84) XOR Key "K" (ASCII 75) = 21 (the key "KEY" repeats).
- Substitution Key (s): "KEY"
- Shift Key (r): 5

The final stage is permutation, where the characters of the XORed data block are permuted by shifting them r positions along the finalized tour path, where r is the Shift Key. That is, the character located at tour position i is moved to the square corresponding to tour position $((i - 1 + r) \bmod m n) + 1$. For instance, consider a simple path of 6 squares and the data "ABCDEF" placed along it. If the Shift Key r is 3, the character 'A' at tour position 1 would move to position $((1 - 1 + 3) \bmod 6) + 1 = 4$. Similarly, 'B' from position 2 would move to position 5, and 'F' from position 6 would move to position $((6 - 1 + 3) \bmod 6) + 1 = 3$. This effectively rotates the characters along the tour path. The final sequence of characters, read from the board from top to bottom and left to right, constitutes the ciphertext.

The decryption process precisely mirrors these steps in the exact reverse order: inverse permutation, then XOR substitution, followed by unpadding by locating the *NULL* delimiter.

Results and Discussion

To evaluate the proposed cryptographic framework, the complete algorithm was implemented and tested using MATLAB Online (R2024b Update 5). It is important to note that, since MATLAB Online is a cloud-based service, the recorded performance metrics are indicative and subject to variation depending on the server's available resources at the time of execution. This section presents the results of that evaluation. First, an illustrative example is provided to demonstrate the step-by-step encryption process. Following this, we present a quantitative analysis covering three key areas: the padding efficiency of the board optimization algorithm (as an empirical verification of Theorem 1), the system's runtime performance, and its security characteristics. The performance benchmarks were conducted using English text samples of various lengths extracted from public-domain literature obtained from Project Gutenberg, ensuring that the test data are representative of a standard text-based application.

Illustrative Example

To demonstrate the complete encryption process, we use the example message "Do you want to build a snowman?". The following keys are used for this example:

First, the plaintext is prepared for encryption. A *NULL* character (`char(0)` in MATLAB) is appended to the original 31-character message to act as an end-of-message delimiter. Thus, the padded text is "Do you want to build a snowman?<NULL>" with a length of 32 characters.

Next, the Optimal Board Search (Algorithm 1) is executed with an input length of 32. An optimal solution found by the algorithm is a 6×6 chessboard, which has 36 squares. This requires $36 - 32 = 4$ characters of additional padding. These are filled with random characters from the original text. For this example, we use "dyau". As a result, the final data block to be encrypted is a 36-character string: "Do you want to build a snowman?<NULL>dyau".

The entire 36-character data block is then encrypted using a Vigenère-style XOR cipher with the Substitution Key "KEY". The key is repeated to match the length of the block ("KEYKEYKEY..."). After that, the padded text is placed along with the repeating key on 6×6 chessboard from left to right and top to bottom, as shown in Figure 2.

D,K	,K	t,K	l,K	n,K	? ,K
o,E	w,E	o,E	d,E	o,E	<NULL>E
,Y	a,Y	,Y	,Y	w,Y	d,Y
y,K	n,K	b,K	a,K	m,K	y,K
o,E	t,E	u,E	,E	a,E	a,E
u,Y	,Y	l,Y	s,Y	n,Y	u,Y

Figure 2. Conceptual layout of the 36-character padded data block paired with the repeating Vigenère key ("KEY"). Each square displays the (data character, key character) pair used as input to the bitwise XOR substitution step

Each character in the data block is XORed with its corresponding character in the repeating key. This transforms our padded message to

'□*y2*,k28%ly?*y)00!y*e%*.*&\$7tE=2\$,.'

Note that the first character '□' is a non-printable character. The result is a 36-character block of intermediate ciphertext, which we will call $C_{intermediate}$. The 36 characters of $C_{intermediate}$ are conceptually placed on the board, as in Figure 3.

□	k	?	'	%	t
*	2	*	!	*	E
y	8	y	y	.	=
2	%	)	*	&	2
*	1	0	e	\$	\$
,	y	0	*	7	,

Figure 3. Intermediate ciphertext placed on the board

The final stage is the permutation of $C_{intermediate}$. The Tour Generation Algorithm (Algorithm 2) is executed on the 6×6 board. It systematically searches and finds a valid closed knight's tour. For consistency, we use the tour path shown in Figure 4.

[23]	[30]	[15]	[4]	[17]	[32]
[14]	[3]	[24]	[31]	[26]	[5]
[29]	[22]	[27]	[16]	[33]	[18]
[2]	[13]	[36]	[25]	[6]	[9]
[21]	[28]	[11]	[8]	[19]	[34]
[12]	[1]	[20]	[35]	[10]	[7]

Figure 4. A 36-step closed knight's tour on a 6×6 board. The number in each square indicates its position in the tour sequence

Then, each character is shifted forward along the tour path by $r = 5$ positions. The character at tour position i is moved to the square corresponding to tour position $((i - 1 + 5) \bmod 36) + 1$. For example, the character at tour position 1 (the character 'y' at square (5,1) in Figure 3) is moved to the square for tour position 6 (the character 'y' at square (3,4) in Figure 5), and the character at tour position 36 (the character ') at square (3,2) in

Figure 3) is moved to the square for tour position 5 (the character 'y' at square (1,5) in Figure 5). After shifting the characters, the final positions are shown in Figure 5.

=	*	7	*	,	y
2	\$	\$	*	*	)
*	%	8	0	1	%
.	e	!	0	y	'
y	□	&	2	*	y
,	t	?	k	E	2

Figure 5. Final character positions on the board after a rotational shift of $r = 5$ positions along the tour path

The final ciphertext is obtained by reading the characters from the final permuted board in Figure 5 in standard order (left to right, top to bottom). This 36-character string is the final encrypted message. The resulting ciphertext, read from Figure 5 is:

'=2*.y,*\$%e□t7\$8!&?*002k,*1y*Ey)%'y2'.

The decryption process follows these steps in the exact reverse order to recover the original message.

Performance Analysis

The algorithm's performance was benchmarked using English text samples of varying lengths extracted from public domain literature. The runtime for finding the optimal board (Algorithm 1), encryption, and decryption was recorded, as shown in Table 1.

Table 1. Performance analysis of the encryption and decryption process for different text lengths. Runtimes for the board optimization search (Algorithm 1) are shown in milliseconds, while encryption and decryption times are in seconds

Text Length (chars)	Board Search (ms)	Encryption (s)	Decryption (s)
100	0.112	0.12	0.13
500	0.091	1.98	1.89
1000	0.096	28.37	19.35
2500	0.107	12.04	12.10
5000	0.129	511.56	526.83

The results shown in Table 1 indicate that the board-finding algorithm (Algorithm 1) is highly efficient. The runtimes for encryption and decryption are nearly identical for most inputs, as expected for a symmetric process. However, the runtime shows a significant non-linear increase for certain inputs. The nature of this performance is particularly evident in the anomalous result for the 1000-character text, which took significantly longer to encrypt (28.37 s) than to decrypt (19.35 s).

This discrepancy is a direct result of the robust board-resizing loop in the methodology. The longer encryption time indicates that the initial optimal board for this text length was a 'difficult' configuration, forcing the algorithm to iteratively find a new, larger board. The decryption process, in contrast, starts with a ciphertext whose length already corresponds to a 'known good' board, allowing it to find the tour much more quickly without this iterative search. This confirms that the primary performance bottleneck is the tour generation process for specific board configurations.

In addition to runtime performance, the padding efficiency of Algorithm 1 was evaluated empirically to validate the bounds established in Theorem 1. The algorithm was executed for 1,000 different text lengths (from $L = 1$ to $L = 1000$), and the number of required dummy characters ($mn - L$) was recorded for each case. Figure 6 presents a histogram of these results.

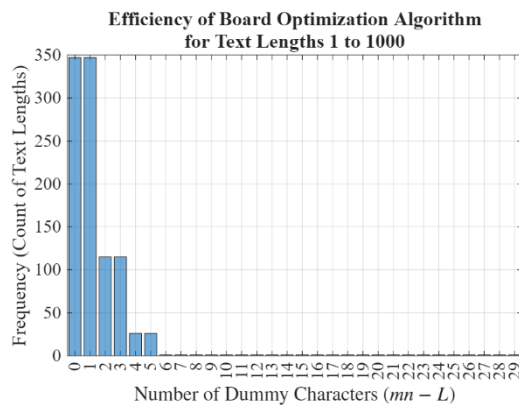


Figure 3. Frequency distribution of the number of dummy characters required for text lengths from 1 to 1,000. The horizontal axis shows the number of dummy characters ($mn - L$), and the vertical axis shows the number of text lengths in that amount of padding

The analysis of the histogram provides two key insights. First, it demonstrates the algorithm's overall efficiency. As shown in the figure, the vast

majority of cases required five or fewer dummy characters, with the most frequent outcomes being 0 or 1. Second, the graph provides a clear visual confirmation of Theorem 1. The few outlier cases that required more than five dummy characters (but fewer than 30) all correspond to input texts where the length L was less than or equal to 30, matching the theorem's predictions.

Security Analysis

The security analysis begins with the keyspace. The security of a cipher against a brute-force attack is determined by the size of its total keyspace. For the proposed algorithm, the keyspace is defined by the possible combinations of the Shift Key (r) and the Substitution Key (s).

The Shift Key r can be any integer from 1 to mn , giving mn choices. The Substitution Key s is an alphanumeric password with a defined maximum length of N characters. Assuming a character set of 62 possibilities (a-z, A-Z, 0-9), the total number of possible passwords is the sum of a geometric series: $62^1 + \dots + 62^N$. Therefore, the total keyspace for an $m \times n$ board is given by:

$$Keyspace = (mn) \times \sum_{i=1}^N 62^i = \frac{62mn}{61} (62^N - 1).$$

For example, for a 6×6 board and a maximum password length of $N = 16$, the keyspace is approximately 2.3×10^{30} . This corresponds to a key length of approximately $\log_2(2.3 \times 10^{30}) \approx 101$ bits. For context, while this is a very large keyspace that makes a brute-force attack computationally infeasible with current technology, it is smaller than the 128-bit keyspace of modern industry standards such as AES-128.

Next, to evaluate the cipher's ability to obscure plaintext statistics, a character frequency analysis was performed. For this test, the first 5,000 characters of the prologue from the Project Gutenberg eBook "The Master Craftsman" by Walter Besant were used as the plaintext sample. The frequency of each byte value (0-255) was counted for both the original plaintext and the resulting ciphertext after encryption. These counts were then normalized to represent the relative frequency of each character, as shown in Figure 7.

The frequency analysis provides strong evidence that the encryption algorithm effectively provides confusion by obscuring the statistical properties of the plaintext. As shown in the figure, the top histogram reveals the predictable, non-uniform distribution of the original plaintext, with distinct peaks for common characters such as the

space (ASCII 32) and lowercase letters (ASCII 97-122). In contrast, the bottom histogram shows that this structure is transformed into a significantly flatter, more uniform pseudorandom distribution in the corresponding ciphertext. This demonstrates that the statistical fingerprint of the original message is successfully hidden, making the cipher more resilient to frequency-analysis attacks.

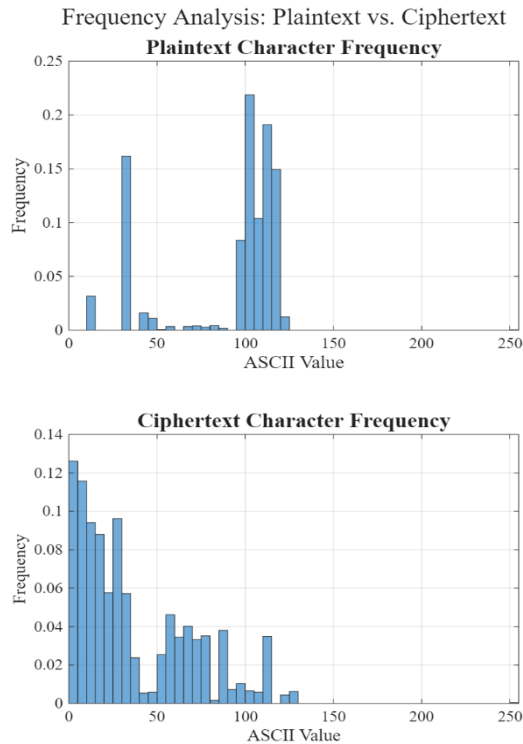


Figure 3. Comparison of character frequency distributions before and after encryption

Lastly, a critical property of a secure cipher is the avalanche effect, which measures the degree of diffusion. It states that a small change in the plaintext or key should result in a drastic, unpredictable change in the ciphertext. For a strong cipher, a single-bit flip in the input should ideally cause approximately 50% of the output bits to flip.

An analysis of the proposed algorithm showed a measured avalanche effect of approximately 0.02% for a 5,000-character message. This low result, while far from the cryptographic ideal, is an expected consequence of the cipher's single-round architecture. The bitwise XOR substitution provides no diffusion, while the knight's tour permutation shuffles the positions of entire characters without mixing bits between them. Consequently, a single-bit change in the input remains contained within a single output character,

preventing the widespread change required for a strong avalanche effect. This analytical finding is the primary motivation for proposing a multi-round architecture with a dedicated bit-level diffusion layer as the key direction for future research.

Discussion

The proposed framework successfully integrates a solution to an optimization problem with a cryptographic application. The primary strength of this work lies in the novel algorithm for dynamically selecting an optimal board size, a unique requirement for encrypting variable-length text with a board-based permutation.

The system is designed for robustness. Its iterative board-resizing mechanism ensures successful encryption for any input text, overcoming the limitations of the Warnsdorff's rule heuristic. Furthermore, as demonstrated by the histogram of dummy characters, Algorithm 1 is highly effective at its primary optimization goal: minimizing wasted space. The trade-off for this reliability and efficiency, however, is performance. The runtime analysis shows that the cryptographic operations can be computationally intensive and do not scale linearly, making the current implementation better suited for messages of moderate length rather than very large data streams.

The security analysis shows a mixed but insightful profile. While the large keyspace and the obfuscation of plaintext statistics are strengths, the poor avalanche effect is a significant weakness. It is important to note, however, that the combination of a polyalphabetic substitution (XOR) with a full-block permutation (knight's tour) makes the cipher resistant to simple known-plaintext attacks. An attacker cannot simply XOR a known plaintext with its ciphertext to recover the key, because the permutation layer decouples the direct relationship between a plaintext character and its final ciphertext position. This resilience underscores the difference between a single-round cipher like this one and modern multi-round ciphers like AES, which are specifically designed to create a strong avalanche effect through repeated applications of substitution and bit-level diffusion.

The design of this system highlights a noteworthy trade-off when using fast heuristics in security applications. The robust iterative resizing mechanism, while ensuring reliability, is the direct cause of the observed performance bottleneck. This demonstrates a conscious design trade-off, prioritizing the reliability of a guaranteed result over the efficiency of predictable performance.

Conclusions

This paper presented a robust algorithmic framework for text encryption that ensures a successful output for any input text. The system's primary limitation is not reliability, but performance scalability and diffusion properties. This analysis leads to several clear directions for future research.

Primarily, to address the performance bottleneck, the current heuristic-based search could be replaced by a complete and linear-time algorithm, such as one based on a divide-and-conquer strategy (Parberry, 1997). This would eliminate the need for the iterative board-resizing loop and significantly improve the system's overall performance. From a security standpoint, the cipher could be enhanced by redesigning it as a multi-round system, applying the XOR-Permutation process repeatedly to strengthen it against various cryptanalytic attacks. To specifically address the poor avalanche effect identified in our analysis, a bit-level diffusion layer, such as one based on matrix multiplication or bit-shuffling operations, should be introduced between these rounds. This would more closely align the cipher's architecture with modern standards. Furthermore, a formal cryptanalysis should be conducted to rigorously evaluate the multi-round cipher's security against differential and linear attacks.

Finally, beyond cryptography, the core algorithmic framework for optimally mapping a 1D data stream onto a 2D constrained grid could be adapted for different data types and applications, such as data watermarking, information hiding, or the procedural generation of patterns where structured permutation is desired.

In conclusion, this work serves as a successful proof of concept and provides a solid foundation for future research into developing more robust and efficient cryptosystems based on these unique mathematical and algorithmic principles.

Funding

The authors received no specific financial support for the research, authorship, and/or publication of this article.

Author Contributions

Conceptualization: R.Y.; Methodology: R.Y.; Software: R.Y.; Formal Analysis: R.Y.; Investigation: R.Y.; Data Curation: R.Y.; Writing - Original Draft: R.Y.; Writing - Review & Editing:

R.Y. The author has read and agreed to the published version of the manuscript.

Ethics Statement

Ethical approval was not required for this study because it did not involve human participants, animals, identifiable personal data, or materials requiring institutional ethical review.

Declaration of Generative AI Use

The authors declare that no Generative AI tools were used in the preparation of this manuscript.

Conflict of Interest

The authors declare that they have no conflicts of interest related to this work.

Data Availability Statement

No new data were generated or analyzed in support of this research.

Acknowledgement

The authors have no acknowledgements to declare.

References

- Chia, G. L., & Ong, S. H. (2005). Generalized knight's tours on rectangular chessboards. *Discrete Applied Mathematics*, 150(1-3), 80-98. <https://doi.org/10.1016/j.dam.2004.11.008>
- Delei, J., Sen, B., & Wenming, D. (2008). An image encryption algorithm based on knight's tour and slip encryption-filter. In *2008 International Conference on Computer Science and Software Engineering* (pp. 251-255). IEEE. <https://doi.org/10.1109/CSSE.2008.1142>
- Demaio, J., & Hippchen, T. (2009). Closed knight's tours with minimal square removal for all rectangular boards. *Mathematics Magazine*, 82(3), 219-225. <https://doi.org/10.1080/0025570X.2009.11953624>
- Gerlach, J. R. (2015). The knight's tour in chess-Implementing a heuristic solution. In *SAS Global Forum 2015 Proceedings*. SAS Institute.
- Gordon, V. S., & Slocum, T. J. (2004). The knight's tour-Evolutionary vs. depth-first search. In *Proceedings of the 2004 Congress on Evolutionary Computation* (pp. 1435-1440). IEEE. <https://doi.org/10.1109/CEC.2004.1331065>
- Khan, M. F., Saleem, K., Shah, T., Hazzazi, M. M., Bahkali, I., & Shukla, P. K. (2022). Block cipher's substitution box generation based on natural randomness in underwater acoustics and knight's tour chain. *Computational Intelligence and Neuroscience*, 2022, Article 8338508. <https://doi.org/10.1155/2022/8338508>

- Lin, S. S., & Wei, C. L. (2005). Optimal algorithms for constructing knight's tours on arbitrary $n \times m$ chessboards. *Discrete Applied Mathematics*, 146(3), 219-232. <https://doi.org/10.1016/j.dam.2004.11.002>
- Mahmood, A. S., & Mohd Rahim, M. S. (2017). Generating and expanding of an encryption key based on knight tour problem. *Journal of Theoretical and Applied Information Technology*, 95(7), 1485-1496. <https://www.jatit.org/volumes/Vol95No7/17Vol95No7.pdf>
- Nie, S. A., Sulong, G., Ali, R., & Abel, A. (2019). The use of least significant bit (LSB) and knight tour algorithm for image steganography of cover image. *International Journal of Electrical and Computer Engineering*, 9(6), 5218-5226. <https://doi.org/10.11591/ijece.v9i6.pp5218-5226>
- Oraby, E. O., & Hamza, R. M. (2023). A modified KLEIN encryption-based knight tour for image encryption. *Journal of Applied Engineering and Technological Science*, 5(1), 268-278. <https://doi.org/10.37385/jaets.v5i1.3296>
- Parberry, I. (1997). An efficient algorithm for the knight's tour problem. *Discrete Applied Mathematics*, 73(3), 251-260. [https://doi.org/10.1016/S0166-218X\(96\)00010-8](https://doi.org/10.1016/S0166-218X(96)00010-8)
- Philip, A. (2013). A generalized pseudo-knight's tour algorithm for encryption of an image. *IEEE Potentials*, 32(6), 10-16. <https://doi.org/10.1109/MPOT.2012.2219651>
- Pohl, I. (1967). A method for finding Hamilton paths and knight's tours. *Communications of the ACM*, 10(7), 446-449. <https://doi.org/10.1145/363427.363463>
- Romanuke, V. V., Yaremko, S. A., Kuzmina, O. M., & Yehoshyna, H. A. (2024). Data scrambler knight tour algorithm. *System Research and Information Technologies*, (3), 44-63. <https://doi.org/10.20535/SRIT.2308-8893.2024.3.03>
- Schwenk, A. J. (1991). Which rectangular chessboards have a knight's tour? *Mathematics Magazine*, 64(5), 325-332. <https://doi.org/10.1080/0025570X.1991.11977627>
- Singh, M., Kakkar, A., & Singh, M. (2015). Image encryption scheme based on knight's tour problem. *Procedia Computer Science*, 70, 245-250. <https://doi.org/10.1016/j.procs.2015.10.081>
- Squirrel, D., & Cull, P. (1996). *A Warnsdorff-rule algorithm for knight's tours* [Unpublished manuscript].
- Yates, C. I., He, J. Y., & Schmidt, N. J. (2025). Comprehensive solutions to variations of the knight's tour problem. *International Journal of High School Research*, 7(3), 71-81.
- Younus, Z. S., & Younus, G. T. (2019). Video steganography using knight tour algorithm and LSB method for encrypted data. *Journal of Intelligent Systems*, 29(1), 1216-1225. <https://doi.org/10.1515/jisys-2018-0225>