A Combined Sudoku and Synthetic Colour Image Techniques for Cryptographic Key Generation

C Manikandan, K Sai Siva Satwik, T M Smarani * and P Umamaheshwari
School of EEE, SASTRA Deemed to be University, Thanjavur 613401, Tamil Nadu, India

tmsmarani@gmail.com

Abstract. The exchange of sensitive digital information has increased significantly in the last decade due to the internet's growth and availability. More robust encryption techniques were introduced to secure the data-in-transit and data-at-rest. The strength of these encryption algorithms depends on the keys. Keys are generated using Pseudorandom Number Generators (PRNG). The key generation algorithm defines these PRNGs. Although there are many robust key generation algorithms, there is a demand for new key generation algorithms to break hackers' experience and knowledge. This paper proposed a new key generation algorithm using value generated from sudoku matrices and synthetic color image. 8x8 sudoku matrices are generated randomly for each session. These sudoku matrices are solved using backtracking and recursive programming. The solved sudoku matrices are used to extract bits from pixels of the synthetic color image. The extracted bits are then combined to form a key. Further, randomness in the key is tested using the NIST (National Institute of Standards and Technology).

1. Introduction

Attackers can observe sensitive information like phone numbers, addresses, medical records, business functions, business secrets, government operations, military operations, and so on in the network. Therefore, data protection is essential when transmitting sensitive information using various digital communication technologies. Sensitive information has to be encrypted before communicating [1]. Many standard algorithms are used for encryption like DES, AES, RSA, RC4, etc. [2-3]. Although the sensitive data is encrypted using any standard algorithms, if the attacker gets the key or guesses the key used for encryption, the encryption algorithm fails to secure the sensitive information. The strength of the encryption algorithm depends on the strength of the key [4]. Therefore, to improve the key's strength, various key generation algorithms have been proposed [5]. The strength of the key depends on the randomness of the key. The proposed work in [6] uses the LSB extraction based key generation technique. The method proposed in [7] uses LSB extraction, absolute difference calculation, and histogram computation to generate the key. In [8], a chaotic logistic map approach is implemented to generate a key from a synthetic color image. All these algorithms have implemented an LSB extraction technique to generate keys for encryption. The methods proposed in [17, 18] uses random scan pattern based key generation techniques using rat-in-a-maze and laser chaos technique. In [15] a dynamic key generation technique is proposed using a combination of logistic and piecewise chaotic maps. The symmetric key encryption algorithm is proposed in [16] using an image as key. NIST has published various documentation related to key strength and how it could affect the security of various encryption techniques. NIST has provided detailed guidance on selecting the keys for different scenarios like encryption, digital signatures, key wrapping, source authentication, passwords,
and so on. It is mentioned that when quantum computing becomes a practical consideration, most of the existing block cipher algorithms fail to meet the security requirements related to key size and key strength. Hence, there is a need for new and complex techniques that generate larger random keys and improve the key strength [14, 19, and 20]. A mini-sudoku matrix-based data embedding scheme is proposed in [9]. In this paper, a novel mini-sudoku-based LSB extraction key generation technique has been proposed for achieving randomness in the key.

Sudoku is a number-placement puzzle consisting of a grid of nine 3-by-3 squares, in which the numbers 1 to 9 must be placed so that each row, column, and block only contains one instance of each number. Within each block, some numbers are provided as clues in certain cells. They appear in various levels of difficulty according to how many numbers are already filled in. Many other variations of sudoku are also possible, like pentomino, heptomino, etc. [10-13]. The proposed work implements 8×8 sudoku matrices, which are solved using backtracking and recursive programming. The sudoku matrices' obtained solutions are used to extract the bits from every synthetic color image pixel. The extracted bits are then combined to form a key. This key is tested for randomness using NIST statistical analysis tool.

The paper is organized as follows. Section 2 provides the methodology to generate sudoku matrices, Section 3 provides the NIST statistical analysis tool results and compares the keyspace and key length of the proposed algorithm with existing algorithms. Section 4 concludes the paper and presents applications and future scope.

2. Methodology
A synthetic color image is generated using three 256×256 grayscale images: red, green, and blue plane images. Each pixel in the grayscale image is represented using 8 bits. Therefore, the bit values vary from 0 to 255. These grayscale images are then combined to form a 256×256 synthetic color image. The pixels in the synthetic color images are random. The generated synthetic color image is shown in Figure 1. No part of the image is repeated within itself, thus making it viable for key generation.

![Figure 1. This synthetic colour image](image.png)

After creating the synthetic color image, random 8×8 sudoku matrices are generated and contain eight 2×4 blocks. These 8×8 sudoku matrices follow the sudoku rules, as mentioned below.

1. All rows must have numbers from 1 to 8 in any order.
2. All columns must have numbers from 1 to 8 in any order.
3. All 2×4 blocks must have numbers from 1 to 8 in any order.
4. No number should be repeated in any row, column, or block.
2.1. Algorithms to solve sudoku matrix

Backtracking is a general technique consisting of rules to solve a computational problem by trying out all possible solutions. This will solve the problems step by step repeatedly to improve the solution. Hence this technique brings out the perfect solution. It removes the responses that don’t satisfy the conditions of the puzzle irrespective of the time. The step by step algorithm to solve the sudoku matrix is given below.

Step 1: Select 8 numbers varying from 1 to 8 and initialize its value as a-h. The integers a-h are declared with a buffer size of 8 bits.
Step 2: Arrange the selected eight numbers at eight positions in an 8×8 sudoku matrix as shown in Figure 2.
Step 3: Assign a number from 1 to 8 at the empty location in the 8×8 sudoku matrix.
Step 4: If the assigned number is equal to any number in the specified row, column, or 2×4, the number is returned.
Step 5: Repeat step 3 and step 4 until a valid number is assigned at the empty location.
Step 6: Select another empty location and repeat the process from step 3 to step 6 until all empty places are filled. This provides the solution for the sudoku matrix, as shown in Figure 3.
Step 7: Rotate the selected eight numbers clockwise and repeat from step 3 to step 7 to solve the sudoku matrix.
Step 8: Repeat step 7 eight times until eight sudoku matrix solutions are generated from one random arrangement
Step 7: Repeat the process from step 2 to step 8 128 times until all sudoku matrices are solved.

![Random sudoku matrix arrangement](image1)

**Figure 2.** Random sudoku matrix arrangement.

![Sample 8×8 sudoku solution](image2)

**Figure 3.** Sample 8×8 sudoku solution
This algorithm provides 1024 sudoku solutions. These sudoku solutions are 8×8 matrices and arranged in an array of size 8192×8. This array is used to extract bits from the synthetic color image to generate a key.

2.2 Algorithm to generate a key

The flow chart represents solving the sudoku matrix and generating the key from a synthetic color image given below in Figure 4. The step by step process of key generation using the proposed algorithm is given below.

**Figure 4.** Flow chart representation of the proposed algorithm

Step 1: Select a 256×256 synthetic color image.
Step 2: Divide the synthetic color image into Red, Green, and Blue plane.
Step 3: Reshape the pixels from 256×256 arrangement into 1×65536 arrangement into an array in all the planes.
Step 4: Generate an 8192×8 array containing sudoku solutions that are generated previously.
Step 5: The numbers in the sudoku solution array represents the bit location of the pixel. For example, a number 5 in a sudoku solution array represents the 5th-bit location from LSB in the pixel.
Step 6: Select the red plane and extract the bit from the corresponding bit location provided by the sudoku solution array in the corresponding pixel.
Step 7: Store the extracted bit in an array. Repeat step 6 for all the pixels in the red plane and store them in an array.
Step 8: Similarly, select the green and blue plane and repeat steps 6 and 7 simultaneously. This step provides three keys of size 65536 bits.
Step 9: XOR three keys to generate one final key of length 65536.
3. Results and Discussion

The algorithm used to generate the key is implemented using sudoku logic. The solved sudoku matrices are used to extract bits from the pixels of the synthetic color image. The sudoku matrices are developed using the C++ programming language and solved as shown in Figure 5, using the proposed backtracking algorithm in C++. The solved sudoku matrices are stored in an 8192×8 array. This array is then stored in an excel file.

![Solved Sudoku Matrix](image)

**Figure 5.** Solved sudoku matrix arrangement.

The key generation algorithm is then implemented using MATLAB. The software developed in MATLAB creates a synthetic color image by using system-defined PRNG. The software then imports the excel file containing solved sudoku matrices. The numbers in sudoku matrices are used as locations of bits in the pixels of the synthetic color image. These bits are then extracted and stored in an array to form a key.

The generated key is tested for randomness using NIST statistical tool. This tool performs various randomness tests like frequency test, runs test, rank test, non-overlapping test, entropy test, serial test, and linear-complexity test. The results are produced in Table 1. It is observed from the results that the key passes all the tests with probability values greater than the threshold (>0.01).

| Test                          | Probability |
|-------------------------------|-------------|
| Frequency                     | 0.4253      |
| Block Frequency               | 0.3164      |
| Cumulative Sums               | 0.3766      |
| Runs                          | 0.3766      |
| Longest Run                   | 0.4529      |
| Rank                          | 0.0376      |
| FFT                           | 0.4253      |
| Non-Overlapping Template      | 0.9932      |
| Approximate Entropy           | 0.3164      |
| Serial                        | 0.9766      |
| Linear Complexity             | 0.2755      |
Table 2. Comparison of keyspace and key length

| Algorithm          | Key length (bits) | KeySpace  |
|--------------------|------------------|-----------|
| LSB Histogram [7]  | 1776             | $2^{1776}$|
| LSB [5]            | 65536            | $2^{65536}$|
| Logistic Map [8]   | 65,536           | $2^{65536}$|
| Sudoku Algorithm   | 65,536           | $2^{65536}$|

The proposed key generation algorithm is compared with the existing key generation algorithm in terms of keyspace and key length. Key length represents the number of bits available in the key, and keyspace represents the total number of attacks required to regenerate the given key. The comparison results are shown in Table 2. It is observed that the proposed algorithm generates 65,536 bits key, which is equivalent to the algorithm proposed in [5, 8] and lesser than algorithms proposed in [7]. This concludes that the proposed algorithm is compatible with various applications like encryption, digital signatures, and so on.

![Stream Cipher Encryption/Decryption](image)

**Figure 6.** Stream Cipher Encryption/Decryption using the generated key.

The key generated using the proposed key generation algorithm is used for stream encryption and decryption. The GUI representation of stream encryption and decryption is shown in Figure 6. Here, the sample plain text is converted into a binary form. The binary format of plain text is XORed with the generated key, and a binary form of ciphertext is generated. On the receiver side, the binary form of ciphertext and key is received. The binary form of ciphertext is XORed with the received key to generate a binary form of plain text. The binary form of plain text is then converted to ASCII code to retrieve sample plain text. Similarly, the proposed key can be used for block encryption, digital signatures, and so on. The efficiency of the proposed key generation algorithm can be observed with the key strength. When the generated key is used in block ciphers, block cipher techniques will be immune to quantum computing attacks. It increases the time taken to crack the generated key due to its massive length of 65,536 bits. When the key generated is used as a private key for digital signatures, it provides verifier impersonation resistance due to the randomness and huge keyspace. Although the other existing algorithms provide similar efficiency, the proposed algorithm's complexity makes it hard for security-sensitive techniques like re-keying and key revocation.
4. Conclusion
This paper proposes a key generation technique by employing a sudoku matrix and synthetic color image. The sudoku matrices are generated and solved using backtracking and recursive programming. The solved matrices are used to extract random bits from the synthetic color image. The extracted bits are used as the key. Later, the generated key is tested for randomness using NIST statistical analysis tool, and results showed that the key is random in nature. The key length and key space analysis are compared with the existing algorithms to determine the key's strength. It is observed that the proposed algorithm meets the standards of the existing algorithms. The key can be used in many applications like encryption, decryption, spreading, and dispreading. Other applications include One Time Passwords (OTP) since a random key is generated each time the algorithm is used. In the future, the proposed algorithm can further be enhanced by using random sudoku patterns to generate sudoku matrices. This further increases the randomness and complexity of the key generation algorithm.

References
[1] Schneier B 2015 Secrets and lies: digital security in a networked world (John Wiley & Sons)
[2] Singh G 2013 Int. J. Comput. Appl. 67
[3] Thambiraja E, Ramesh G and Umarani D R 2012. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 2(26)
[4] Katz J and Lindell Y 2014 Introduction to modern cryptography (CRC Press)
[5] Manikandan C, Neelamegam P, Kumar R, Babu V and Sai Siva Satwik K 2019 Int. J. Comput. Netw. Commun. 1115
[6] Barhoom T S and Abusilmiyeh Z M 2013 Proc. Int. Conf. on Information and Communication Technology(Gaza) vol 1(New York: IEEE) pp 71–76
[7] Riddhi N and Gamit N 2015 Proc. Int. Conf. on Computer, Communication and Control (Indore) vol 1(New York: IEEE) pp 1–5
[8] Manikandan C, Kumar S R, Nikhith K, Gayathri M S and Neelamegam P 2019 Proc. Int. Conf. on Applications and Techniques in Information Security (Singapore) vol 1(New York: Springer) pp 142–155.
[9] He M, Liu Y, Chang CC and He M 2019 IEEE Access, 7141414
[10] Aaronson L 2006 Sudoku Science (New York: IEEE vol 43) pp 16–17
[11] Berthier D 2007 The Hidden Logic of Sudoku (Raleigh, NC: Lulu. com) p 76
[12] Lei L and Fuke S 2007 Comput. Knowl. Tech 2 481
[13] Chipantasi D J M and Erazo N R V 2015 Proc. Int. Conf. on Image and Graphics (Cham) vol 1(New York: Springer) pp 603–613
[14] Chen L, Jordan S, Liu Y K, Moody D, Peralta R, Perlne R. and Smith-Tone D 2016 Report on Post-Quantum Cryptography Tech. Rep. 2 NISTIR 8105 (Gaithersburg: NIST)
[15] Jawad L M and Sulong G 2015 Mod. Appl. Sci 9 85.
[16] Islam M, Shah M, Khan Z, Mahmood T and Khan M J 2015 Proc. Int. Conf. on Frontiers of Information Technology (Islamabad) vol 1(New York: IEEE) pp 1–5
[17] Sivakumar T and Li P 2019 Opt. Laser. Technol. 111 196
[18] Sivakumar T and Anusha T 2016 Asian J. Inf. Technol. 15 4635
[19] Waltermire D and Scarfone K 2011 Guide to using vulnerability naming schemes. NIST Special Publication, SP 800-51 Rev. 1(Gaithersburg: NIST)
[20] Barker E and Roginsky A 2018 Transitioning the use of cryptographic algorithms and key lengths NIST Special Publication (SP) 800-131A Rev. 2 (Gaithersburg: NIST)