Open Access
2020, 17(2):567-574

Baghdad Science Journal

P-ISSN: 2078-8665
E-ISSN: 2411-7986

DOI: http://dx.doi.org/10.21123/bsj.2020.17.2.0567

# A Cryptosystem for Database Security Based on TSFS Algorithm

*Saad A. Abdulameer*[1 *] *Ali H. Kashmar*[2] *Ammar I. Shihab*[2]

Received 23/3/2019, Accepted 3/11/2019, Published 1/6/2020

**Abstract:**

Implementation of TSFS (Transposition, Substitution, Folding, and Shifting) algorithm as an encryption algorithm in database security had limitations in character set and the number of keys used. The proposed cryptosystem is based on making some enhancements on the phases of TSFS encryption algorithm by computing the determinant of the keys matrices which affects the implementation of the algorithm phases. These changes showed high security to the database against different types of security attacks by achieving both goals of confusion and diffusion.

**Key words:** Cryptosystem, Database, Security, TSFS.

## Introduction:

Numerous business fields have computerized the majority of their information and applications (1). Unauthorized users may access the system, either for copying that information or rolling out malicious actions to part or the whole database. The security mechanism of a database management system (DBMS) must adopt a security technique of information encryption to ensure the protection of sensitive information (2).

Sensitive information may represent medicinal records, master card numbers, and government secret data. Thus the database administrators should guarantee appropriate insurance of sensitive information. Database cryptosystems can be portrayed by the accompanying properties: confidentiality, integrity, and availability (3).

Basically TSFS constructs of (plain-text - encryption algorithm - encryption key – cipher-text - decryption algorithm - decryption key) as illustrated in Fig. 1 (4).

[1] College of Education for Women, University of Baghdad, Baghdad, Iraq.
[2]Computer Science Department, College of Science, University of Baghdad, Baghdad, Iraq.
[*]Corresponding author: saadoodi2003@gmail.com
[*]ORCID ID: https://orcid.org/0000-0002-9644-3388

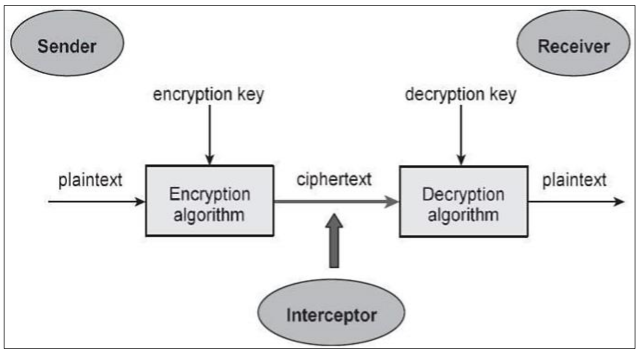


**Figure 1. The components of cryptosystem (4)**

TSFS is a symmetric algorithm (the same keys are used in encryption and decryption). It encrypts a block of data as a plain-text or message *m* under the activity of a secret key *k* to produce cipher-text *c*. This is defined as:

$$c = \mathrm{Enc}_k(m) \rightarrow \text{Equation 1}$$

And reversed by the decryption process with the user-supplied key that is used in encryption, and defined as (5):

$$m = \mathrm{Dec}_k(c) \rightarrow \text{Equation 2}$$

**Related Work:**

Manivannan and Sttjarani (6) suggested TSFS as an encryption algorithm by using the symmetric key, which incorporates transpositions and substitutions as highlights in the procedures that reduces the processing time of encryption and decryption. Afterward proposed an enhanced work on TSFS which can scramble the information that consists of alphanumeric and couple of special characters guaranteeing abnormal state of security to encrypt information however forces a couple of imperatives on the information size and the special characters utilized. So, if someone wants to encrypt the user email id, for example, it is not possible to encrypt by using this algorithm.

Al-Souly *et al*. (7) proposed TSFS algorithm by dividing the dataset to isolate characters, amending shifting and substitution processes, supporting a couple of modulo factors and 4×16 arrays correspondingly to maintain a strategic distance from the incorrectness that emerges in the decryption steps. Initially, a random key generator has been utilized just for obtaining the key values. At that point, the key is replaced by numbers dependent on the succession in the alphabet and saved in 4×4 matrix form. The expansion tasks for the keys are achieved by exchanging positions of the columns utilizing add round key technique. It was a comparative study with well-established encryption algorithms DES (Data Encryption Standard) and AES (Advanced Encryption Standard). So, due to time constraints, the algorithm did not cover all special symbols.

Preeti and Khatkar (8) aimed to improve the TSFS algorithm and to give powerful protection to the databases while restricting the additional cost of encryption and decryption time by encoding sensitive information as it were. The ETSFS algorithm can scramble the information that comprises of all numbers, alphabetic characters from A to Z, and some symbols: (*, - . /: @ and _). The improved TSFS algorithm is symmetric, which means every procedure or transformation must be reversible and have a backwards process that can drop its impact. The key additionally should be utilized in an inverse manner. Again, the algorithm in this study did not cover all special symbols.

Jawanjal *et al.* (9) declared that the significant objective is to try the following issues:

1. Performing an enhancement of TSFS algorithm that will uphold special characters too.
2. A lightweight algorithm is proposed, so it will require minimal time to scramble information.
3. Main key1, key 2 which are arbitrarily utilized in ETSFS may have numbers.
4. Development of an ETSFS Algorithm in WampServer version 2.5 and PHP to estimate.
5. Examination of security by ETSFS Algorithm.

Still, the same number of characters and symbols that were allowed in the previous studies to be encrypted data. Ganta and Zhou (10) made few improvements to the existing algorithm LWSE (Light Weight Symmetric Encryption):

1. Unlimited size for input data.
2. For each time the algorithm executes different random keys generated.
3. It applies an XOR operation on a final output with the key k1. This is denoted as OTP (One Time Pad) algorithm.

The limitation of this algorithm is the character dataset includes (0-25) alphabets, (0-9) numbers, and (0-29) special characters.

**Problem Statement**

Most of cryptosystems based on TSFS algorithm that are built so far (mentioned in the related work) have the following properties:

- The encryption performed inside the DBMS which provides insufficient security against intruders, malicious users, or corrupted database administrators.
- The limitation of character set used for either the input text (plaintext) or the encryption keys. The character set used included alphanumeric characters of the ASCII code (alphabetic characters from A to Z and numerical characters from 0 to 9) with some special characters that is used in IP addressing and email (*, -, ., /, :, @, and _) that forms in all of 43 characters.
- Beside the encryption keys, there was no other factor affects the encryption (and decryption) algorithm's path in a way that append some novelty.

**The Objectives**

1. To design a cryptosystem based on TSFS algorithm for database security.
2. To increase the character set used in the encryption process for both plaintext and encryption keys.
3. To develop a strong and high performance of encryption keys generated through the use of genetic algorithm.
4. To add sort of modernity to the TSFS algorithm by calculating the determinant of all keys matrices to be a decision factors for using different techniques throughout the algorithm path.

**Proposed Cryptosystem:**

The proposed cryptosystem implements TSFS algorithm on plain-text taken from SQLite database through encryption process to produced cipher-text which will be stored in a text file and then by decryption process converts that cipher-text back to plain-text as shown in Fig. 2.

**Structure Design:**

TSFS algorithm takes plain-text in the shape of a 4 × 4 matrix of any printable characters shown in Table 1 that are ranged from 32-126 ASCII characters (95 characters), any character that is not within the range of the printable set remains un-encrypted. Padding is also used if the input data is less than 16 characters to add ' ' space character (Its ASCII is 32) at the end of the plain-text.

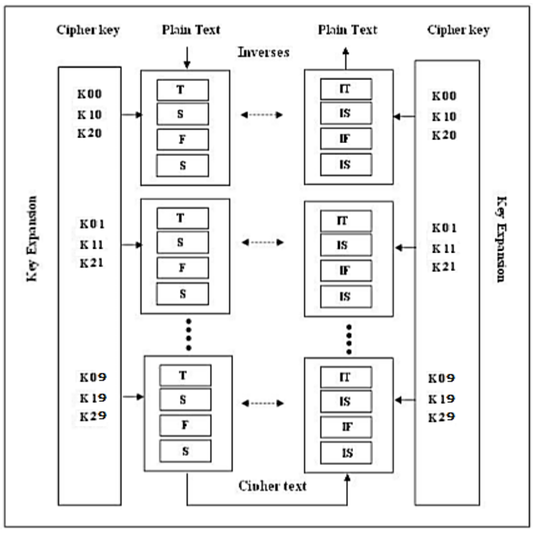


**Figure 2. TSFS Algorithm of the proposed cryptosystem**

**Table 1. ASCII Table – Printable Characters**

| Character | Hex | Decimal | Character | Hex | Decimal | Character | Hex | Decimal |
|---|---|---|---|---|---|---|---|---|
| | 20 | 32 | @ | 40 | 64 | ` | 60 | 96 |
| ! | 21 | 33 | A | 41 | 65 | a | 61 | 97 |
| " | 22 | 34 | B | 42 | 66 | b | 62 | 98 |
| # | 23 | 35 | C | 43 | 67 | c | 63 | 99 |
| $ | 24 | 36 | D | 44 | 68 | d | 64 | 100 |
| % | 25 | 37 | E | 45 | 69 | e | 65 | 101 |
| & | 26 | 38 | F | 46 | 70 | f | 66 | 102 |
| ' | 27 | 39 | G | 47 | 71 | g | 67 | 103 |
| ( | 28 | 40 | H | 48 | 72 | h | 68 | 104 |
| ) | 29 | 41 | I | 49 | 73 | i | 69 | 105 |
| * | 2a | 42 | J | 4a | 74 | j | 6a | 106 |
| + | 2b | 43 | K | 4b | 75 | k | 6b | 107 |
| , | 2c | 44 | L | 4c | 76 | l | 6c | 108 |
| - | 2d | 45 | M | 4d | 77 | m | 6d | 109 |
| . | 2e | 46 | N | 4e | 78 | n | 6e | 110 |
| / | 2f | 47 | O | 4f | 79 | o | 6f | 111 |
| 0 | 30 | 48 | P | 50 | 80 | p | 70 | 112 |
| 1 | 31 | 49 | Q | 51 | 81 | q | 71 | 113 |
| 2 | 32 | 50 | R | 52 | 82 | r | 72 | 114 |
| 3 | 33 | 51 | S | 53 | 83 | s | 73 | 115 |
| 4 | 34 | 52 | T | 54 | 84 | t | 74 | 116 |
| 5 | 35 | 53 | U | 55 | 85 | u | 75 | 117 |
| 6 | 36 | 54 | V | 56 | 86 | v | 76 | 118 |
| 7 | 37 | 55 | W | 57 | 87 | w | 77 | 119 |
| 8 | 38 | 56 | X | 58 | 88 | x | 78 | 120 |
| 9 | 39 | 57 | Y | 59 | 89 | y | 79 | 121 |
| : | 3a | 58 | Z | 5a | 90 | z | 7a | 122 |
| ; | 3b | 59 | [ | 5b | 91 | { | 7b | 123 |
| < | 3c | 60 | \ | 5c | 92 | \| | 7c | 124 |
| = | 3d | 61 | ] | 5d | 93 | } | 7d | 125 |
| > | 3e | 62 | ^ | 5e | 94 | ~ | 7e | 126 |
| ? | 3f | 63 | _ | 5f | 95 | Delete | 7f | 127 |

**Keys Generation and Expansion:**

Three keys each of 16 printable characters (128 bits length) are randomly generated using Genetic Algorithm (11) (as shown in Fig. 3) at the beginning of each encryption process that represents k00, k10, and k20.

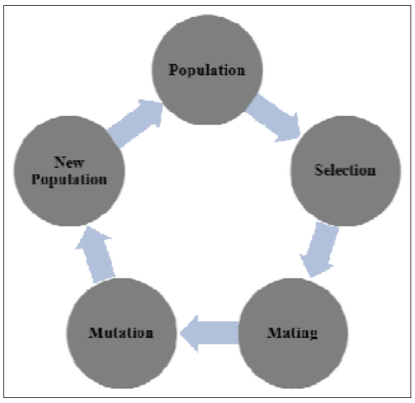
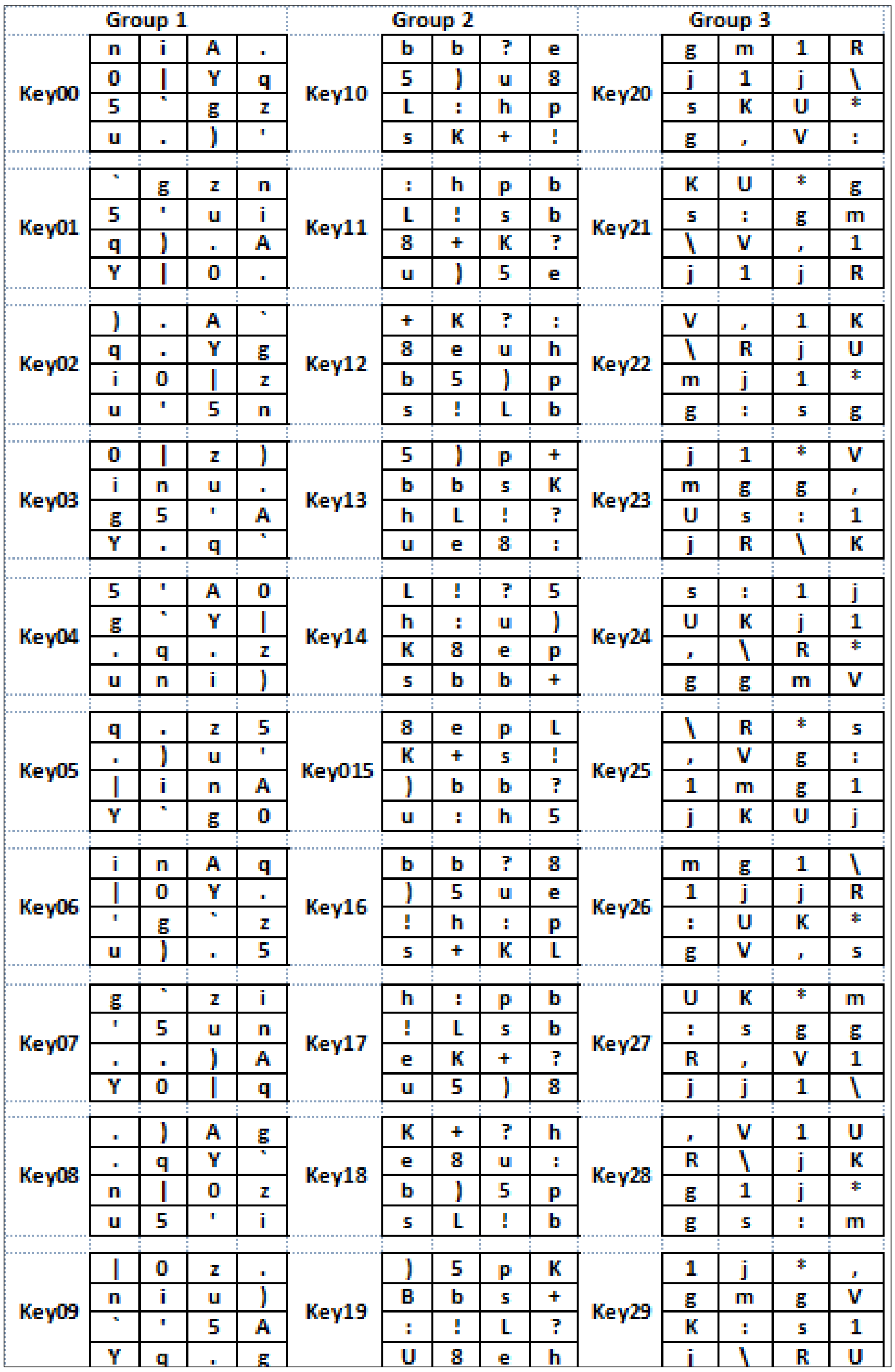


**Figure 3. Basic Model of Genetic Algorithm** (12)

The key generation using the Genetic Algorithm will experience various procedures and primary criteria for key selection will depend on the fitness value of the population, these procedures insure randomness of the keys generated as illustrated in the following algorithm:

```
Do
    Step one
        Selection: To breed a new generation selected is a portion of the existing population.
    Step two
        Crossover: An operation of choosing solutions of more than one parent to produce a child solution.
    Step three
        Mutation: Changes one gene value (or more) in a chromosome from its former state.
Until getting all elements of the key matrix. (13)
```

Each of the three keys generated is expanded into nine keys by performing Route Transposition (14) of the initial key and subsequently resulted keys to gather three groups as illustrated in the example of Fig. 4.



| Group 1 | | | | | Group 2 | | | | | Group 3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Key00 | n | i | A | . | Key10 | b | b | ? | e | Key20 | g | m | 1 | R |
| | 0 | \| | Y | q | | 5 | ) | u | 8 | | j | 1 | j | \ |
| | 5 | ` | g | z | | L | : | h | p | | s | K | U | * |
| | u | . | ) | ' | | s | K | + | ! | | g | , | V | : |
| Key01 | ` | g | z | n | Key11 | : | h | p | b | Key21 | K | U | * | g |
| | 5 | ' | u | i | | L | ! | s | b | | s | : | g | m |
| | q | ) | . | A | | 8 | + | K | ? | | \ | V | , | 1 |
| | Y | \| | 0 | . | | u | ) | 5 | e | | j | 1 | j | R |
| Key02 | ) | . | A | ` | Key12 | + | K | ? | : | Key22 | V | , | 1 | K |
| | q | . | Y | g | | 8 | e | u | h | | \ | R | j | U |
| | i | 0 | \| | z | | b | 5 | ) | p | | m | j | 1 | * |
| | u | ' | 5 | n | | s | ! | L | b | | g | : | s | g |
| Key03 | 0 | \| | z | ) | Key13 | 5 | ) | p | + | Key23 | j | 1 | * | V |
| | i | n | u | . | | b | b | s | K | | m | g | g | , |
| | g | 5 | ' | A | | h | L | ! | ? | | U | s | : | 1 |
| | Y | . | q | ` | | u | e | 8 | : | | j | R | \ | K |
| Key04 | 5 | ' | A | 0 | Key14 | L | ! | ? | 5 | Key24 | s | : | 1 | j |
| | g | ` | Y | \| | | h | : | u | ) | | U | K | j | 1 |
| | . | q | . | z | | K | 8 | e | p | | , | \ | R | * |
| | u | n | i | ) | | s | b | b | + | | g | g | m | V |
| Key05 | q | . | z | 5 | Key015 | 8 | e | p | L | Key25 | \ | R | * | s |
| | . | ) | u | ' | | K | + | s | ! | | , | V | g | : |
| | \| | i | n | A | | ) | b | b | ? | | 1 | m | g | 1 |
| | Y | ` | g | 0 | | u | : | h | 5 | | j | K | U | j |
| Key06 | i | n | A | q | Key16 | b | b | ? | 8 | Key26 | m | g | 1 | \ |
| | \| | 0 | Y | . | | ) | 5 | u | e | | 1 | j | j | R |
| | ' | g | ` | z | | ! | h | : | p | | : | U | K | * |
| | u | ) | . | 5 | | s | + | K | L | | g | V | , | s |
| Key07 | g | ` | z | i | Key17 | h | : | p | b | Key27 | U | K | * | m |
| | ' | 5 | u | n | | ! | L | s | b | | : | s | g | g |
| | . | . | ) | A | | e | K | + | ? | | R | , | V | 1 |
| | Y | 0 | \| | q | | u | 5 | ) | 8 | | j | j | 1 | \ |
| Key08 | . | ) | A | g | Key18 | K | + | ? | h | Key28 | , | V | 1 | U |
| | . | q | Y | ` | | e | 8 | u | : | | R | \ | j | K |
| | n | \| | 0 | z | | b | ) | 5 | p | | g | 1 | j | * |
| | u | 5 | ' | i | | s | L | ! | b | | g | s | : | m |
| Key09 | \| | 0 | z | . | Key19 | ) | 5 | p | K | Key29 | 1 | j | * | , |
| | n | i | u | ) | | B | b | s | + | | g | m | g | V |
| | ` | ' | 5 | A | | : | ! | L | ? | | K | : | s | 1 |
| | Y | q | . | g | | U | 8 | e | h | | j | \ | R | U |

**Figure 4. Example of key expansion**

**The Determinant** (15)**:**

For each square matrix, there is a unique element called the determinant of that matrix. Over the 10 rounds of TSFS algorithm, the maximum unsigned integer part of the determinant of three keys from each group is calculated from the ASCII of each character in the key string as the equation 6:

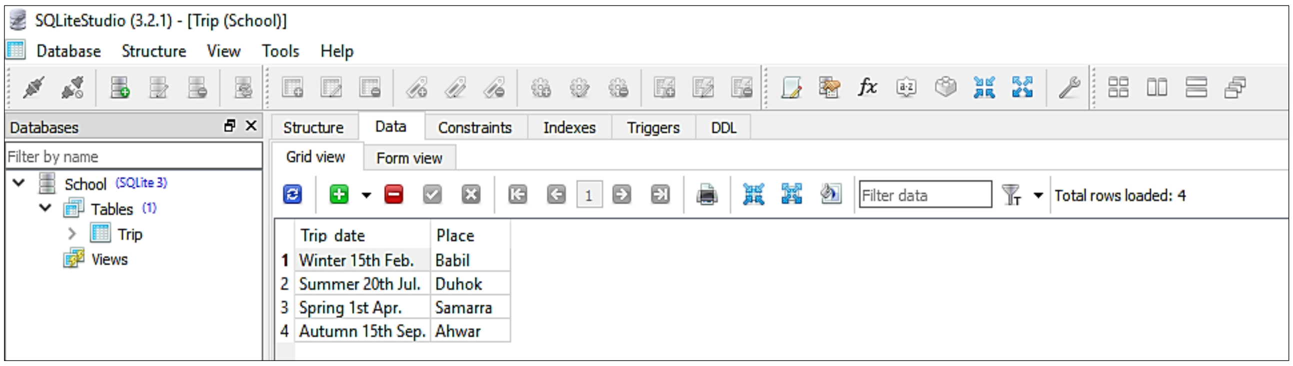
Det0 = (Det(key00) mod 95) + 32→ Equation 3
Det1 = (Det(key10) mod 95) + 32→ Equation 4
Det2 = (Det(key20) mod 95) + 32→ Equation 5
MAXDet = MAX(Det0, Det1, Det2)→ Equation 6

If all of the three determinants are equal then the one of a key from the first group is chosen.

For the example above, the determinants of the initial keys will be:
Det0 = 116 Det1 = 117 Det2 = 104
Then Det1 that represents Group1 is selected to be MAXDet of the first round. This factor will play a great role in Transposition and Folding phases, as will be shown, which represents an enhancement to the TSFS algorithm that increased the strength of the encryption and decryption processes.

**Methodology:**

TSFS algorithm implemented in its encryption process on SQLite database on its rest (stored on some media and no read/write operations are performed while using the cryptosystem). As an experimental database, School.db will be chosen which includes one table named (Trip) of four records of the fields illustrated in Fig. 5:


SQLiteStudio (3.2.1) - [Trip (School)]
Database Structure View Tools Help
Databases
Structure Data Constraints Indexes Triggers DDL
Grid view Form view
Filter by name
School (SQLite 3)
Tables (1)
Trip
Views
Filter data
Total rows loaded: 4

| | Trip date | Place |
|---|---|---|
| 1 | Winter 15th Feb. | Babil |
| 2 | Summer 20th Jul. | Duhok |
| 3 | Spring 1st Apr. | Samarra |
| 4 | Autumn 15th Sep. | Ahwar |

**Figure 5. Experimental Database**

The plain-text is read from a SQLite database and entered to the TSFS algorithm in blocks, each block size of $4 \times 4$ matrix. The first block is 'Winter 15th Feb.', and by using the three keys that are generated in the example above and computing their Det and the MAXDet for the first round (of the ten rounds), then the encryption process of the algorithm is illustrated in Fig. 6:

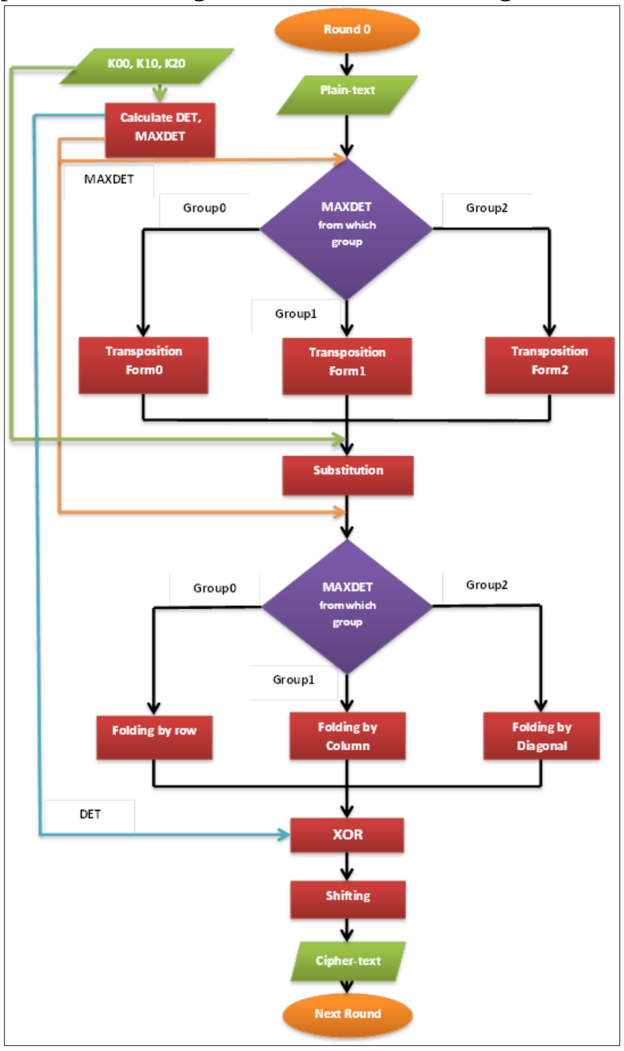
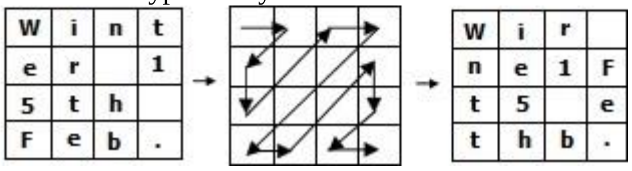


**Figure 6. Encryption Process of TSFS Algorithm**

**Transposition:**

TSFS algorithm uses three directions of zigzag diagonal transposition that rearrange matrix elements. The element in the first position of the plain-text is placed in different positions of the transposition-ciphered matrix. The use of one of the three directions depends on the MAXDet of that round. Transposition cipher is performed as follows:

1) If the MAXDet is from Group0 of the encryption keys:

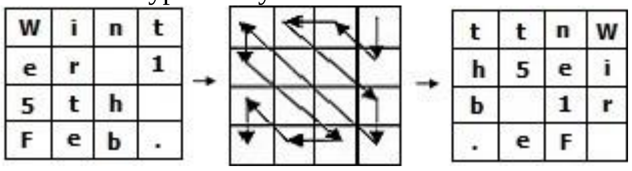


| | | | |
|---|---|---|---|
| W | i | n | t |
| e | r | | 1 |
| 5 | t | h | |
| F | e | b | . |

| | | | |
|---|---|---|---|
| W | i | r | |
| n | e | 1 | F |
| t | 5 | | e |
| t | h | b | . |

**Figure 7. Zigzag diagonal transposition form 1**

2) If the MAXDet is from Group1 of the encryption keys:

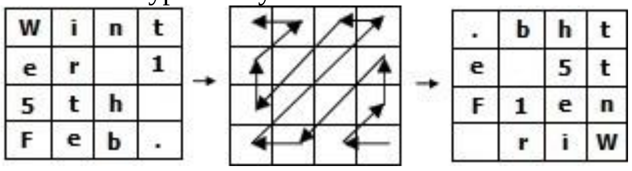


| | | | |
|---|---|---|---|
| W | i | n | t |
| e | r | | 1 |
| 5 | t | h | |
| F | e | b | . |

| | | | |
|---|---|---|---|
| t | t | n | W |
| h | 5 | e | i |
| b | | 1 | r |
| . | e | F | |

**Figure 8. Zigzag diagonal transposition form 2**

3) If the MAXDet is from Group2 of the encryption keys:



| | | | |
|---|---|---|---|
| W | i | n | t |
| e | r | | 1 |
| 5 | t | h | |
| F | e | b | . |

| | | | |
|---|---|---|---|
| . | b | h | t |
| e | | 5 | t |
| F | 1 | e | n |
| | r | i | W |

**Figure 9. Zigzag diagonal transposition form 3**

This procedure is used for both encryption and decryption phases.

**Substitution:**

It is the replacement process of plain-text elements with other elements from the same character set by using three keys in each round. The encryption equation for any given element x is:

E(x) = ((((((p + k1) mod 95) + k2) mod 95) + k3) mod 95) + 32 → Equation 7

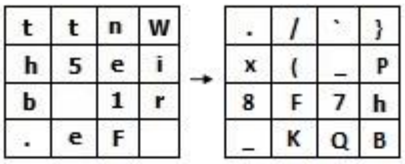
The decryption equation D is:

D(E(x)) = ((((((p - k1) mod 95) - k2) mod 95) - k3) mod 95) + 32 → Equation 8

If the decryption value is less than 0 then D will be:

D = 95 – D → Equation 9

Proceeding to the example above, at the first round k00, k10, and k20 are considered with E equation with the plain-text resulted from transposition technique to produce substitution ciphered matrix:



| | | | |
|---|---|---|---|
| t | t | n | W |
| h | 5 | e | i |
| b | | 1 | r |
| . | e | F | |

| | | | |
|---|---|---|---|
| . | / | ` | } |
| x | ( | _ | P |
| 8 | F | 7 | h |
| _ | K | Q | B |

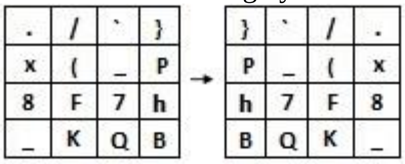
**Figure 10. Substitution phase**

**Folding:**

The matrix resulted from the substitution technique is folded like a paper and the direction of folding depends on the MAXDet of that round which will be:

1) Folding by row if the MAXDet is from Group0.
2) Folding by column if the MAXDet is from Group1.
3) Folding diagonally if the MAXDet is from Group2.

In the example, since the MAXDet was from Group1 then it must be folding by column:



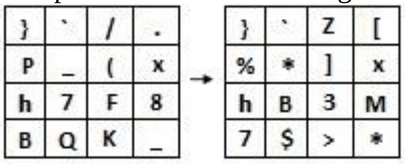
| | | | |
|---|---|---|---|
| . | / | ` | } |
| x | ( | _ | P |
| 8 | F | 7 | h |
| _ | K | Q | B |

| | | | |
|---|---|---|---|
| } | ` | / | . |
| P | _ | ( | x |
| h | 7 | F | 8 |
| B | Q | K | _ |

**Figure 11. Folding by column**

Then the MAXDet value of that round is XORed to all elements of the resulted matrix only if the result of XOR operation remains within the range of the character set (32-126).

So Det1 which is equal to 117 in the example will produce the matrix in Fig. 12:



| | | | |
|---|---|---|---|
| } | ` | / | . |
| P | _ | ( | x |
| h | 7 | F | 8 |
| B | Q | K | _ |

| | | | |
|---|---|---|---|
| } | ` | Z | [ |
| % | * | ] | x |
| h | B | 3 | M |
| 7 | $ | > | * |

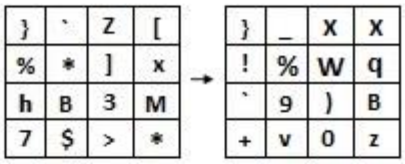
**Figure 12. MAXDet XOR folding matrix**

In the decryption phase, MAXDet is XORed to the matrix resulted from previous procedure then folded according to MAXDet position rules stated above.

**Shifting:**

The last technique in the methodology that replaces each element with its corresponding element located in the character set backward by its position in the folded matrix.

The letter (Z) in position 2 of the resulted folded matrix of the example, then its ASCII number (90) is shifted backward two places to be the ASCII of (88) which represent the (X) letter, and so on for all elements taking in concern that the shifting is circulated within the range of the character set as illustrated in Fig. 13. Decryption phase is shifted forward.



| | | | |
|---|---|---|---|
| } | ` | Z | [ |
| % | * | ] | x |
| h | B | 3 | M |
| 7 | $ | > | * |

| | | | |
|---|---|---|---|
| } | _ | X | X |
| ! | % | W | q |
| ` | 9 | ) | B |
| + | v | 0 | z |

**Figure 13. Shifting phase**

## Results and Discussion:

The cipher-text resulted from the encryption process is stored in a text file called (School_Trip.txt) and to be used in the decryption process to retrieve to the original data:

PXlXO ~ITcIzS`GPrDnsupM|T%4T&p8_
PicoO ~1TbIhS@G"rDnsuC}|TY1T&-I_
Q-_3O%Z17cI}S G:hD"s;L]ER&7KSK&V
PX<:OY%#TcIxS`G"rDnsub)|T0_T&KM_

This cipher-text is complex enough to stand against brute-force attack that will need $2^{128}$ alternations to guess each of the initial 3 randomly generated keys (16):

| | |
|---|---|
| K00 = | niA.0IYq5`gzu.)' |
| K10 = | bb?e5)u8L:hpsK+! |
| K20 = | gm1Rj1j\sKU*g,V: |

These keys are stored in a separated file called (School_Trip.key). The complexity of the cipher-text also comes from the use of the determinant factor that is changed with each run of the cryptosystem and with each round of the 10 rounds of the TSFS algorithm.

The latency for the encryption process = 0:00:00.053387 ms which is concerned fair enough regarding the cost of computation time.

## Conclusions:

Database attacks are incremented in the risks of information revelation. Numerous organizations must have an arrangement with regulation and legislation on information confidentiality.

The best arrangement focused on protecting the information is utilizing cryptography, alongside different techniques. In the event that sensitive information is encrypted in the database, risks from security faults can be wiped out by using three cryptographic keys for keeping the information protected. This paper proposed a cryptosystem based on the TSFS algorithm which if compared with previously proposed cryptosystems based on the same algorithm will be advanced with the following features:

1. A powerful cryptosystem for database security.
2. Enhancing the use of the four techniques of the TSFS (Transposition Substitution Folding Shifting) algorithm to encrypt/decrypt large amount of data.
3. The dependence on the GA (Genetic Algorithm) as an assistant algorithm ensures the generation of random encryption keys.
4. Increasing the number of characters to be encrypted and the characters that compose the encryption keys to involve all printable characters that are ranged from 32-126 of ASCII code.
5. Using outside the database encryption keeps the original database without any change and produces different files for the transmission and storing purposes. The decryption process produces a copy of the original database.
6. Invulnerability against different types of attacks due to the complexity of the cipher-text.
7. Low computation cost due to the well-designed cryptosystem and the considerably small latency of encryption and decryption processes.

It is recommended to enhance this cryptosystem as follows:

1. Making use of this cryptosystem in the security of other data types like documents, excel sheets, images, media files, and so on.
2. Increasing the data block size to $8 \times 8$ or even $16 \times 16$ characters (bytes) instead of $4 \times 4$ characters and consequently the size of the keys to decrease the computation time.
3. Involving more characters in the dataset used by the algorithm to include other than English language like Arabic language characters for example.
4. Using the threading technique in programming the cryptosystem since each block of the data is encrypted/decrypted independently of the others.

## Authors' declaration:

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are mine ours. Besides, the Figures and images, which are not mine ours, have been given the permission for re-publication attached with the manuscript.
- Ethical Clearance: The project was approved by the local ethical committee in University of Baghdad.

## References:

1. Thuraisingham B. Database and Applications Security: Integrating Information Security and Data Management. Boca Raton. FL: Auerbach Publications. 2005.
2. Chávez J. Basic Principles of Database Security. Universidad Politécnica Territorial del estado Aragua. 2015.
3. Paraskevopoulou Z, Giannarakis N. Database Security & Cryptography. National Technical University of Athens, School of Electrical and Computer Engineering. 2013.
4. Stallings W. Cryptography and Network Security, 4th Ed. Principles and Practices, Prentice Hall. 2005.
5. Knudsen L, Robshaw M. The Block Cipher Companion. Springer Publishing Company, Incorporated. 2011.
6. Manivannan D, Sttjarani R. Light weight and secure database encryption using TSFS algorithm. Proceedings of the International Conference on Computing Communication and Networking Technologies. 2010.
7. Al-Souly H, Al-Sheddi A, Kurdi H. Lightweight Symmetric Encryption Algorithm for Secure Database. Science and Information Conference. 2013.
8. Preeti, Khatkar K. Enhancing Data Security and Privacy on Web OS Using TSFS. International Journal of Advanced Research in Computer and Communication Engineering JARCCE. 2015 August; 4(8).
9. Jawanjal N, Tijare P, Sawalkar S. Implementing ETSFS Algorithm for Database Security. International Research Journal of Engineering and Technology IRJET. 2017 May; 04(05).
10. Ganta K, Zhou B. Combined Enhanced LWSE Algorithm with OTP Algorithm for Secure Database. International Journal of Computers and Communications IJCNC. 2017; 11.
11. https://colindrake.me/post/a-genetic-algorithm-in-python.
12. Soni A, Agrawal S. Key Generation Using Genetic Algorithm for Image Encryption. International Journal of Computer Science and Mobile Computing. IJCSMC. 2013 June; 2(6).
13. Habeeb SH, Hassan R. Sensors data encryption using TSFS Algorithm. Journal of Madent Alelem College. 2018; 10(1).
14. http://www.crypto-it.net/eng/simple/route-cipher.html.
15. Joyce D. Determinants, part III Math 130 Linear Algebra. Clark University. Fall 2015.
16. Barakat M, Eder C, Hanke T. An Introduction to Cryptography. Lecture notes at the University of Kaiserslautern, the summer term 2017.

# نظام تشفير لحماية قاعدة البيانات باستخدام خوارزمية TSFS

سعد عبد الكريم عبد الامير[1] علي حبيب كشمر[2] عمار ابراهيم شهاب[2]

[1] كلية التربية للبنات، جامعة بغداد، بغداد، العراق
[2] قسم الحاسوب، كلية العلوم، جامعة بغداد، بغداد، العراق

**الخلاصة:**
تعاني خوارزمية TSFS (التحويل، الاستبدال، الطي، النقل) من تحديد في مجموعة الاحرف والرموز المستخدمة وعدد مفاتيح التشفير. نظام التشفير المقترح يستند الى اجراء بعض التحسينات وذلك بحساب مقرر مصفوفات مفاتيح التشفير والذي سيؤثر في تطبيق العمليات الاربعة للخوارزمية. هذه التغييرات اظهرت نتائج عالية في حماية قواعد البيانات ضد مختلف انواع الانتهاكات وذلك من خلال تحقيق هدفي التشويش والانتشار في عملية التشفير.

**الكلمات المفتاحية:** نظام تشفير، قاعدة بيانات، أمنية، TSFS