New Transport Layer Security using Metaheuristics and New Key Exchange Protocol

Mohamed Kaddouri[1], Mohammed Bouhdadi[2]
LMPHE Laboratory Mohammed V University,
Faculty of Sciences Rabat
Rabat, Morocco. BP1014 RP

Zakaria Kaddouri[3], Driss Guerchi[4], Bouchra Echandouri[5]
Department of Computer Science,
Mohammed V University Abu Dhabi
Abu Dhabi, United Arab Emirates. P.O. 106621

***Abstract*—The easiness of data transmission is one of the information security flaws that needs to be handled rigorously. It makes eavesdropping, tampering and message forgery by malicious more simple. One of the protocols developed to secure communication between the client and the server consists of using Transport Layer Security (TLS). TLS is a cryptographic protocol that allows encryption using record protocol, authentication and data integrity. In this paper, a new TLS version is proposed, named Transport Layer Security with Metaheuristics (TLSM), which is based on a recently designed metaheuristic symmetric ciphering technique for data encryption, combined with hash function SHA-SBOX and a new method for private key exchange. Compared to the existing TLS versions, the suggested protocol outperform all of them in terms of level of security of the encrypted data, key management and execution time.**

Keywords*—Transport Layer Security (TLS); metaheuristic; symmetric ciphering algorithm; private key exchange; hash function

I. Introduction

Computer communication involving online transactions and payment in exchange for goods and services grew over the last decade drastically. However, an untrustworthy e-commerce website makes the costumers not having sufficient trust. One of the protocols that is widely employed to secure these transactions by providing authentication and encryption is the Transport Layer Security (TLS) protocol [12]. In this protocol, the encryption is used to prevent the interception of sensitive data, such as credit card numbers and account passwords.

TLS is a cryptographic protocol that provides end-to-end secure communication for different types of applications [12]. It was adopted by the IETF and specified as an RFC standard. It is the most widely used protocol between a client and a server. It is composed of two main parts: the Handshake Protocol and the Record Protocol [1]. The Handshake Protocol is in charge of key establishment employed to cipher data in the Record Protocol [13]. The simplicity of the handshake is due to its use of public key cryptography, which allows the negotiation of a shared secret key over a risky channel and without prior knowledge between the client and the server.

As a part of the TLS, the handshake protocol also allows the authentication of the presumed identity. The Record Protocol ensures a secure channel for the management of data delivery. The TLS protocol also provides its own data framing and authenticating method [12]. Despite all these security measures, the last version of the TLS protocol has suffered recently from several attacks [11, 14]. Namely, Denial of Service Attacks that attempt to saturate the server with SYN queries during the Handshake. Also, there are other attacks, such as SWEET32 (CVE-2016-2183, CVE-2016-6329)[15], DROWN (CVE-2016-0800)[16] and POODLE (CVE-2014-0160) [17], that are mainly related to weaknesses detected during the encryption.

According to the recent work presented in [2], several Bleichenbacher oracle attacks have shown some vulnerabilities in the TLS1.3 since they succeeded in the cryptanalysis of the RSA encrypted message [3].

The TLS uses RSA to exchange the secret key. This key is shared and decided by the client, then encrypted by the (server) public key before being sent to the server [1]. However, the process of key exchange using RSA [3] has several drawbacks, such as the slowness of any new connection [14].

In order to increase the speed of new connections, in this paper the use of the recently developed symmetrical metaheuristics ciphering technique [4] is suggested in the most sensitive phase of TLS protocol. This technique helps secure messages and private keys (session keys) exchanged between the client and the server. The simulation results show better execution time performance and higher security robustness.

The rest of this paper is organized as follows: in Section 2, related work are presented. Then in Section 3, the proposed solution TLSM are describe in details. Section 4 discusses the performance evaluation and security analysis of this approach. The last section concludes this paper and presents future works.

II. Related work

As mentioned in the introduction, TLS is a well-known and employed cryptographic protocol. The majority of security protocols do not use a mechanism about how to distribute secret keys. However, asymmetric cryptography, in particular RSA, has been declared much resource consuming for limited devices. Further, the last proposed versions of TLS suffer from some vulnerabilities that lead to attack success [2].

In Eronen et al. paper [5], the authors specified a set of cipher-suits for the Transport Layer Security protocol that support symmetric pre-shared keys based authentication. These processes are very slow because of the use of the Diffie-Hellman key exchange in authentication with the pre-shared key. The server and the client are authenticated with asymmetric encryption using pre-shared key.

The work Sizzle [6], a tinny version of TLS, was improved using Elliptic Curve Cryptography public key cryptosystem. It helps secure an end to end communication for devices with tight computational memory and energy constraints. However, it loses some security in its process.

Furthermore, in the paper [7], the authors show weaknesses of TLS. They presented a recovery attack against TLS when RC4 is employed in encryption. Their attack is based on the statistical analysis of RC4.

III. New Transport Layer Security Protocol using Metaheuristics

A. Background

1) Transport Layer Security : Over the past years, the Internet Engineering Task Force has been continuously working on developing one of its most important security protocols: the Transport Layer Security (TLS). It is a cryptographic protocol that ensures web security through encryption and authenticity of https website. It is supported by mainly two protocols: The Handshake Protocol and the Record Protocol [1].

The Handshake Protocol is conceived to negotiate a key security between a client and server. The handshake process is started by the client sending the message: $'ClientHello'$ including a random number and cipher suites. To respond to this message, the server sends a random number, a chosen cipher suit and a server certificate. The server's certificate is then authenticated by the client who generates a secret key. This secret key is encrypted before transmission by the client using the server public key. Once received by the server, the encrypted secret key is decrypted and used in the subsequent encryption steps [8].

The Record Protocol guarantees a secure channel for the management of data delivery. It starts by splitting the data into series of blocks. Afterwards, these blocks are compressed. Then, a message authentication code (MAC) is applied to these compressed blocks using the shared secret key. Thereafter, this block is encrypted with a symmetric encryption algorithm [8].

2) Symmetrical Metaheuristics Ciphering Approach : Symmetrical metaheuristics ciphering is a new cipher system using Vigenere and metaheuristics [4]. The aim of the use of Vigenere encryption is to maximize the confusion and the use of metaheuristics helps generate a robust secret Meta-key.

At the beginning of the encryption process, Vigenere algorithm is applied. Thereafter, a random key is generated. The ASCII number of every key's character will be added. The initial cipher (Vigenere cipher) is created and is given in table. Thereafter to every value is assigned a list of coordinates of the plain value. Shuffling begins by permutation, then the best solution is chosen using metaheuristic algorithms and evaluated by the evaluation function.

The final cipher is formed using the new list and each value is assigned to the coordinates stored in the beginning.

3) SHA-SBOX: SHA-SBOX [9] is an iterated hash function, inspired from SHA hash function, where the compression function involves permutation and substitution. Further, the Boolean functions Ch and Ma, used in SHA family, have been replaced by the substitution and permutation functions FPS. This FPS function takes three blocks of 32 bits, concatenates them and splits them into two blocks of 48 bits each. The two 48-bit blocks are then subject to permutations P1 and P2, respectively [9].

A substitution functions are then applied to the two 48-bit blocks producing hence two blocks of 32 bits each. The substitution functions are the well-known S-boxes provided in the DES encryption algorithm to ensure a good confusion. Thereafter, a modular addition is applied on the two 32-bit blocks.

B. TLSM Description

1) TLSM handshake protocol: The proposed TLSM protocol consists of several steps (Fig. 1). It uses both asymmetric and symmetric encryptions. The client and the server begin with a negotiation of the employed algorithms and the adopted an exchange of the key. In the handshake protocol, which is the most essential phase of establishing a secure connection, the server and the client exchange information are used to define the connection properties.

a. Step 1: Client Hello In the first step, the client sends a $\backslash ClientHello$" message to the server. This message includes the following list of information:
   - Client Version of the TLSM with the sequence of supported algorithms.
   - Client Time-Date, a 4-byte date representing the current date and time of the client (in epoch format).
   - Session ID employed for the connection. The server searches for previous sessions, if this session is not empty, it remains in the current session.
   - Compression methods employed for compressing TLSM packets.
   - Cipher Suites (the combinations of cryptographic methods). It contains one cryptographic algorithm for namely: key exchange, data encryption and authentication. The proposed TLSM cipher suite is: $TLSM\_ECDHE\_ECDSA\_WITH\_SMC\_SHASBOX$. It consists of the following information:
     - TLSM :Transport Layer Security with Metaheuristics.
     - ECDHE (Elliptic curve Diffie–Hellman): indicates the key exchange algorithm being employed.
     - ECDSA (Elliptic Curve Digital Signature Algorithm): indicates the authentication algorithm being employed.
     - SMC (Symmetrical Metaheuristic Ciphering): the ciphering algorithm.
     - SHA-SBOX: indicates the message authentication algorithm that is used to authenticate a message.
     - Compression methods: TLS compression methods, the data will be compressed before ciphering it.

b. Step 2: Server Hello The server replies with a $\backslash ServerHello$" when it receives $\backslash ClientHello$"

from the client side, that contains the proposed and selected options during the "$\backslash ClientHello$" or a failure message.

- Server Version is TLSM protocol.
- Server Time-date indicates the current date and time of the client.
- Session ID serves for a new session and resume sessions already open.
- Cipher Suites. The server employs the Cipher Suites sent in the "$\backslash ClientHello$".
- Compression Methods. The server will use the Compression Methods sent in the "$\backslash ClientHello$".

c. Step 3: Server Certificate The server, at this step, sends a signed TLSM certificate containing its public key to prove its identity to the client.

d. Step 4: Client Certificate (Optional) The client must provide his signed certificate, in case of the server asks the client to be authenticated with his certificate.

e. Step 5: Server Key Exchange This message will be sent to the server in case the certificate provided by the server is not sufficient for the client to exchange the symmetrical encryption Key.

f. Step 6: Server Hello Done This message is be sent to the client to affirm that the "$\backslash ServerHello$" message is completed.

g. Step 7: Client Key Exchange Once the "$\backslash HelloDone$" message is received from the server, the Client send its "$\backslash KeyExchange$" message. Then, the "$\backslash ClientKeyExchange$" is sent in case the server asks for a client certificate. Thereafter, the client generates the Meta-secret key. It is worth mentioning that before transmitting the Meta-secret key to the server, the client ciphers this key using the server's public key. The asymmetric encryption is employed for the Meta-secret key exchange. Once the server receives the Meta-secret key, it employs its private key to decrypt it. Thereafter, for the rest of the communication, the latter is used by both the client and the server to encrypt and decrypt the data respectively.

h. Step 8: Client Change Cipher Spec Once the "$\backslash ClientKeyExchange$" is finished all data communication between the client and the server is secure. The protocol "$\backslash ChangeCipherSpec$" is employed to change the encryption.

i. Step 9: Client Handshake Finished This message is sent when the server receives the last message of the handshake process from the client.

j. Step 10: Server Change Cipher Spec All data sent communication, at this step, in the server side is secure.

k. Step 11: Server Handshake Finished This message will be sent when the client receives the last message of the handshake process from the server.

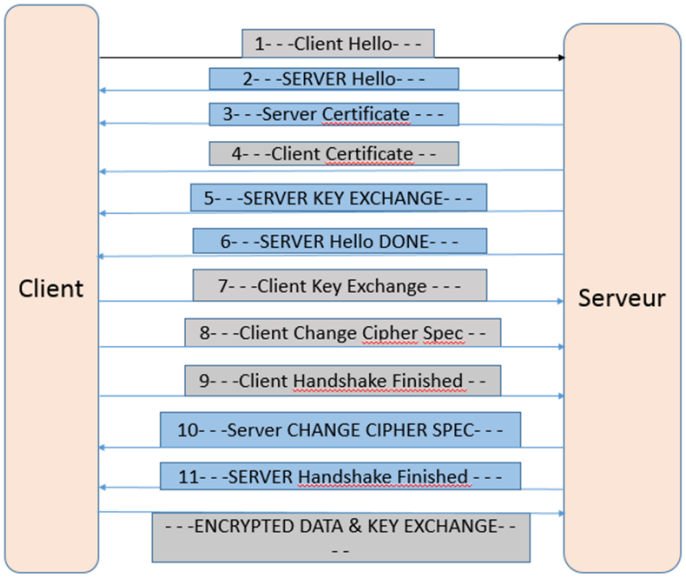

Fig. 1. TLSM Handshake Protocol.

2) TLSM record protocol : The TLSM consists of the following steps (Fig. 2):

a. Step 1 (Data): Application data blocks are split to several blocks with the same size.

b. Step 2 (Fragment): The first fragment is subject to compression, producing the output C(F1).

c. Step 3 (Hash Function): the digit termed E(F1) of C(F1) is calculated using the SHA-SBOX hash function [9].

d. Step 4 (Symmetric Encrypt & Meta-key Exchange): E(F1) is appended at the tail of C(F1), the result is encrypted by the symmetrical metaheuristics ciphering [4] using the first meta-key obtained from the handshake protocol , hence resulting in the encrypted data.

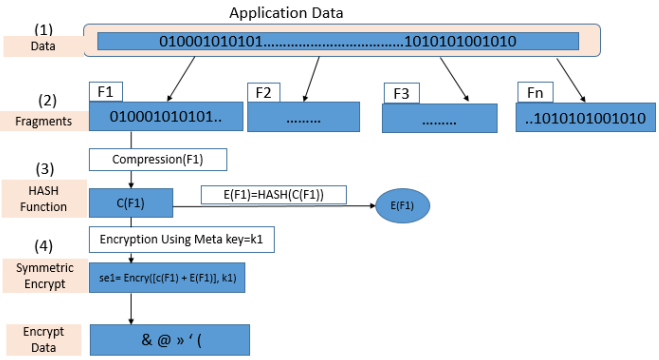

Fig. 2. TLSM record protocol.

3) TLSM Key Exchange and Encryption Process : Before presenting the operations of the key exchange and encryption processes, the following adopted parameters should be defined:

- N: length of the whole data

- N_f: length of each data frame F
- L_{km}: length (in bits) of the Metaheuristic key $k_{m,i}$ generated by the proposed algorithm for the data group i
- L_{sk} : length (in bits) of the sub-key sk_i of the future meta-key
- M: number of data frames per group

Hence the total number of groups N_g (which is the number of keys) is equal to

$$N_g = round(M * \frac{L_{sb}}{L_{km}}) \quad (1)$$

where M = $\frac{N}{N_f}$ is the overall number of frames. The following algorithm illustrates the different steps involved in the encryption and decryption phases (Fig. 3):

Encryption Phase and Key Exchange (Client Side)

- Use the Symmetrical Metaheuristic ciphering to generate the first metaheuristic key $k_{m,1}$.
- Subdivide $k_{m,1}$ into sub-keys sk_i of the same size $(k_{m,1} = [sk_{1,1}, sk_{2,1}, ..., sk_{M,1}])$.

For j = 1: N_g, If j = 1 (First data group)

For i = 1: M,

- the sub-key $sk_{i,j}$ to the end of the data frame $F_{i,j}$: this results in a composite data frame $[F_{i,j}, sk_{i,j}]$
- Encrypt the composite data frame $[F_i, sk_i]$ using the metaheuristic key k_H obtained in the handshake protocol, this results in an encrypted frame $EF_{i,j} = encrypt([F_{i,j}, sk_{i,j}], k_H)$

End Else For i = 1:M,

- Append the sub-key $sk_{i,j}$ to the end of the data frame $F_{i,j}$: this results in a composite data frame $[F_{i,j}, sk_{i,j}]$
- Encrypt the composite data frame $[F_{i,j}, sk_{i,j}]$ using the metaheuristic key $k_{m,j-1}$, this results in an encrypted frame $EF_{i,j} = encrypt([F_{i,j}, sk_{i,j}], k_{m,j-1})$

End

End

- Use the Symmetrical Metaheuristic ciphering to generate the metaheuristic key $k_{m,j+1}$ for the next data group.
- Subdivide $k_{m,j+1}$ into sub-keys sk_i of the same size $k_{m,j+1} = [sk_1, sk_2, ..., sk_M])$.

End

Decryption Phase and Keys Reproduction (Side Side) For j = 1: N_g, If j = 1 (First data group) For i = 1: M,

- Decrypt the received composite data frame $EF_{i,j}$ using the metaheuristic key k_H obtained in the handshake protocol, this results in an decrypted frame $[F_i, sk_{i,j}] = decrypt([EF_{i,j}, k_H)$
- Decompose the decrypted frame $[F_{i,j}, sk_{i,j}]$ into two parts: $F_{i,j}$ and $sk_{i,j}$

End Concatenate all the frames F_i into one data group. Concatenate all the sub-keys $sk_{i,j}$ into one key km,1 Else For i = 1: M,

- Decrypt the received composite data frame $EF_{i,j}$ using the metaheuristic key $k_{m,j-1}$, this results in an decrypted frame $[F_{i,j}, sk_{i,j}] = decrypt(EF_{i,j}, k_{m,j-1})$
- Decompose the decrypted frame $[F_{i,j}, sk_{i,j}]$ into two parts: $F_{i,j}$ and $sk_{i,j}$

End Concatenate all the frames $F_{i,j}$ into one data group. Concatenate all the sub-keys $sk_{i,j}$ into one key $k_{m,j}$ End

End

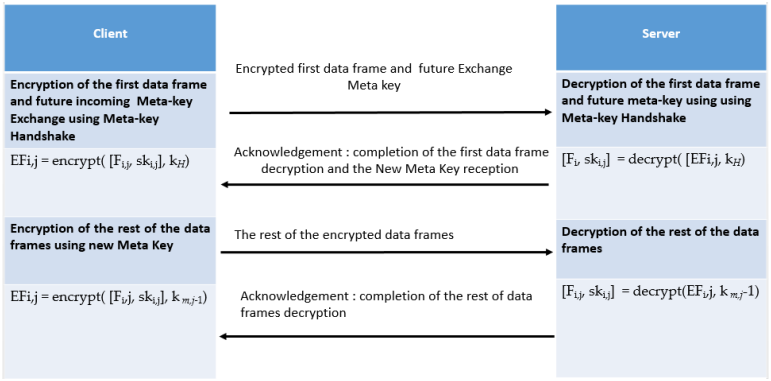

Fig. 3. Key exchange and encryption process.

IV. Security Analysis and Performance

The main purpose of this section is to evaluate the robustness of the proposed TLSM protocol.

A. Symmetrical Metaheuristics Ciphering Execution Time

In the following Table I, in detail, the execution time of Symmetrical metaheuristics ciphering compared to 3DES, AES is presented. These results prove that Symmetrical metaheuristics ciphering needs less time to encrypt the different blocks compared to the required time to be encrypted by the other encryption standards.

TABLE I. Symmetrical Metaheuristics Ciphering Execution Time.

Block size (bytes)	AES	3DES	SMC
16020	1.62	2.18	0.69
34280	2.66	2.43	0.65
72620	3.78	6.15	1.34
157440	6.31	10.90	2.52
224116	7.20	12.56	3.93

B. Sha-Sbox Robustness

Sha-sbox is a hash function designed by the combination of SHA-256 and DES [9]. The compression function is based on permutation and substitution (S_BOX). Sha-sbox has a good avalanche effect [10] (i.e. if one bit changed in the initial message input, it affects the half size of the output). Sha-sbox[9] is efficient, fast, resistant to attack and resist to differential and linear cryptanalysis. Its Security level against birthday attack is 2^{128} bits. Furthermore, Sha-sbox[9] uses in its process uses Fps operations that are not costly in execution time. The modular addition applied to the output from permutations and substitution at every round, increases the level of security and avoid collision attack.

C. Meta Secret Key Security Analysis

Symmetrical metaheuristics ciphering [4] generates keys valid for only one session. For this matter, to encrypt a data frame using Symmetrical metaheuristics ciphering in t sessions, k_t keys are required to be generated. Since the robustness of an encryption algorithm is related to the robustness of its secret key, the probability to find the right key P_A is the maximum number of keys K_N by the expression (2):

$$P_A = \frac{1}{K_N} \tag{2}$$

Where K_N denotes the maximum number of different keys generated by the Symmetrical metaheuristics ciphering. Denote by P_A the probability to find the right key using the brute-force attack. Since in Symmetrical metaheuristics, ciphering each data frame is allocated by 8 bits and each data frame of length N samples is encrypted using one meta-secret key. This meta-secret key size is defined by $8N$ bits (Table II).

TABLE II. KEYS SIZE AND BRUTE-FORCE ATTACK COMPLEXITY.

Frame Length	P_A	K_N	Key Size (in bit)	Brute force attack
40	$1.23e-48$	40!	320	2320
80	$1.40e-119$	80!	640	2640
160	$2.12e-285$	160!	1280	21280
320	$4.73e-665$	320!	2560	22560
640	$1.55e-1520$	640!	5120	25120

It is worth to mention that the shortest meta-secret key generated (320 bits) is better than the current AES key 256-bit which is the actual standard for encryption. It is hard to break this meta-secret key using the brute force attack.

D. Key Session Security Analysis

The protection provided by a symmetric encryption algorithm is related to the length of the key that is expressed in bits. In fact, the length of the key quantifies the maximum number of operations needed to find the right key. It is therefore an essential point for the security of the system. The proposed algorithm generates a number of keys in one session that depends on four parameters the size of data exchanged, the size of the processed blocks of data, the sub-blocks size of the future meta-key and the meta-key size. The results in Table III obtained using the general formula (1). Table III shows the number of keys N_g generated for different data lengths (in bits) and different sub-key sk_i of the future meta-key as following. Here a data frame length N_f of 1000 Bytes and a Metaheuristic key $k_{m,i}$ of 320 bits are choosen.

TABLE III. MAXIMUM NUMBER OF KEYS GENERATED BY THE SYMMETRICAL METAHEURISTICS CIPHERING, K_N.

	16020	34280	72620	157440	224116
40	2	4	8	16	25
80	4	7	15	32	45
160	7	14	30	63	90
320	13	28	59	126	180
640	27	55	117	252	359

The metaheuristic encryption system in its initial version has proven its resistance against most known attacks [11]. We have proposed in this paper a new technique for generating and sharing symmetric keys in the data encryption phase. The TLSM record protocol allows generating an additional number of symmetric keys as the data are processed in the encryption phase. This number depends on the size of the processed data and other parameters that are related to the future meta-keys. In Table III a data size of 224116 bytes is combined with 320-bit encryption keys, data frame size of 1000 bytes and 640-bit sub-keys. With these parameters the system can generate up to 359 different meta-keys, a number that is relatively huge. In addition to the basic security of the SMC encryption system, the new proposed technique increases the security of the Protocol record and hence the overall security of the TLSM Protocol while compared to the standard versions of the TLS Protocol.

V. CONCLUSION

We presented in this paper a new protocol termed TLSM to improve the end-to-end secure communication. It uses a previously developed Metaheuristic symmetric ciphering algorithm in order to encrypt data, a recently presented hash function SHA-SBOX and on a new method for private key exchange. The use of these new algorithms in TLS process made the proposed protocol fast and secure as proven by the simulation results. Providing a high performance compared to TLS previous version, this proposed protocol TLSM proved to be robust and not execution-time consuming. In order to continuously improve the TLS protocol and the new TLSM version proposed in this article, it is focused on the cryptanalysis of the encryption suites used in the handshake protocol. In case of vulnerability found or needed improvement, it is suggested to try to design other encryption suites alternatives more robust and performing to ensuring maximum security of data communication.

REFERENCES

[1] Krawczyk, Hugo, Kenneth G. Paterson, and Hoeteck Wee. *On the security of the TLS protocol: A systematic analysis.* Advances in Cryptology–CRYPTO 2013. Springer, Berlin, Heidelberg, 2013. 429-448.

[2] Ronen, E., Gillham, R., Genkin, D., Shamir, A., Wong, D., Yarom, Y. *The 9 Lives of Bleichenbacher's CAT: New Cache ATtacks on TLS Implementations.*2019

[3] Jonsson, Jakob, and Burton S. Kaliski. *On the Security of RSA Encryption in TLS.* Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 2002, p. 127-142

[4] Kaddouri, Zakaria, Mohamed Amine Hyaya, and Mohamed Kaddouri. *A New Cryptosystem using Vigenere and Metaheuristics for RGB Pixel Shuffling.* Coordinates 25.4 2017: 255.

[5] Eronen, Pasi, and Hannes Tschofenig. *Pre-shared key ciphersuites for transport layer security (TLS)*. No. RFC 4279. 2005.

[6] Gupta, Vipul and Wurm, Michael and Zhu, Yu and Millard, Matthew and Fung, Stephen and Gura, Nils and Eberle, Hans and Shantz, Sheueling Chang, *Sizzle: A standards-based end-to-end security architecture for the embedded internet*, 2005, Sun Microsystems, Inc.

[7] Fardan, N., Bernstein, D. J., Paterson, K. G., Poettering, B., Schuldt, J. C. *On the Security of RC4 in TLS.* In Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13). 2013. (pp. 305-320).

[8] Handa, Arun. *System engineering for IMS networks.* Newnes, 2009.

[9] Zakaria Kaddouri, Fouzia Omary, Abdollah Abouchouar And Mohssin Daari *New Compression Function To Sha-256 Based On The Techniques Of Des*, Journal of Theoretical and Applied Information Technology, 2013, Volume 5,pages 230 – 234.

[10] Castro, J. C. H., Sierra, J. M., Seznec, A., Izquierdo, A., Ribagorda, A. *The strict avalanche criterion randomness test.* Mathematics and Computers in Simulation, 2005, 68(1), 1-7.

[11] Paterson, K. G., Ristenpart, T., Shrimpton, T. *Tag size does matter: Attacks and proofs for the TLS record protocol.* In International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg. (2011, December). (pp. 372-389).

[12] Paulson, Lawrence C. *Inductive analysis of the Internet protocol TLS.* ACM Transactions on Information and System Security (TISSEC) 2.3 (1999): 332-351.

[13] Jiao, R., Ouyang, H., Lin, Y., Luo, Y., Li, G., Jiang, Z., Zheng, Q. *A Computation-Efficient Group Key Distribution Protocol Based on a New Secret Sharing Scheme.* Information, (2019). 10(5), 175.

[14] Chen, H. P., Gonzalez, E., Saez, Y., Kish, L. *Cable capacitance attack against the KLJN secure key exchange.* Information, (2015). 6(4), 719-732.

[15] STANEK, Martin. *Secure by default-the case of TLS.* arXiv preprint arXiv:1708.07569, 2017.

[16] AVIRAM, Nimrod, SCHINZEL, Sebastian, SOMOROVSKY, Juraj, et al. *DROWN: Breaking TLS Using SSLv2.* In : 25th USENIX Security Symposium (USENIX Security 16). 2016. p. 689-706.

[17] SHEFFER, Yaron, HOLZ, Ralph, et SAINT-ANDRE, Peter. *Summarizing known attacks on transport layer security (TLS) and datagram TLS (DTLS).* 2015.