



# Reduce the Memory Used in Key Management for Security Systems

Yasser Ali Alahmadi

Department of Computer Science, Sheba Region University, Marib, Yemen  
yasser\_ali8891@yahoo.com

Mokhtar Alsorori

Department of Computer Science, Sheba Region University, Marib, Yemen  
msorori201201@gmail.com

Saleh Noman Alassali

Department of Computer Science, Sheba Region University, Marib, Yemen  
sano2010123@gmail.com

Received: 05 July 2021 / Revised: 07 August 2021 / Accepted: 10 August 2021 / Published: 28 August 2021

**Abstract** – Nowadays, most of the applications are distributed and require two or more parties to establish a secure communication channel over an open network. Key management is one of the major security issues in such applications. A good security system should reduce more complex problems related to the proper key management and secure-saving of a little number of secret keys at every endpoint. So it is difficult to save one key secretly, and the difficulty will be more and more if the number of secret keys increased. In the literature, many schemes have been proposed for key distribution and management. Although, such schemes have reduced the number of secret keys stored at the users to only one key, Key Distribution Center (KDC), known here as Key Managing Center (KMC), still maintains a shared secret key with each user in the network. In this paper, we propose a method to reduce the number of secret keys stored at the KDC to only one key, regardless of the network size. In the proposed method, the KMC will store a unique stuff data for every user. The user's secret key will be generated by taking the stuff data, adding the lifetime of the secret key, and then hashing the resulting string using the manager secret key. The output digest will be used as the user's secret key. By this way, KMC will only store one key called the manager secret key. Furthermore, we will combine the proposed method with our previous work to build an efficient key management model. Analysis and experimental results indicate that the developed model is highly secure, practical and efficient.

**Index Terms** – Key Management, Key Distribution, Key Storage, Public Key Cryptography, Symmetric Key Cryptography, Formal Verification.

## 1. INTRODUCTION

In an environment consisting of a large-scale communication network that requires every pair of users to establish a secure communication channel and agree on a shared secret key, if symmetric key cryptography is being used, then key

management can become a critical problem[1-4]. In such a network every pair of users requires a shared secret key to secure their communications. Thus every user needs to store  $(N - 1)$  secret keys, where  $N$  is the total number of users in the network. To eliminate this problem, the solution is to employ a trusted third party called Key Distribution Center (KDC)[5], where all users share their secret keys with the KDC rather than share a secret key between every pair of users[6].

In the literature, there are several key distribution and management schemes based on KDC such as [7-18]. In most of these schemes, every user shares a unique secret key with the KDC for purposes of key distribution. If two users wish to communicate with each other in a secure manner, they can obtain a secret session key from the KDC which in turn generates a session key and encrypts it using the secret keys that it shares with those users, and then sends it to them.

### 1.1. Problem Description

There is a difficulty in the process of storing and distributing keys in security systems based on symmetric key cryptography, especially in a large computer network with millions of users. This difficulty is, the KDC has more than one party who want to communicate and exchange information securely. In most existing key distribution and management schemes, the KDC stores  $N$  secret keys in its database for all users in the network and the number of secret keys increases as the number of users increased. Therefore, such schemes suffer from high key storage at the KDC. Moreover, if the KDC is compromised, then all the secret keys will be exposed to the attacker[19, 20]. Thus the secret key update process must be performed for all users in the network. The costs associated with updating these keys are

**RESEARCH ARTICLE**

painfully high in terms of time, effort, financial resources[21]. In addition, the re-establishment of such keys with all users requires out-of-band transmission or through a secure channel.

### 1.2. Objectives

The important objectives of this paper are:

- To facilitate process of saving keys in a secure manner.
- To generate the required keys and securely distribute them to the valid users involved in the network.

These objectives will be achieved by building a software key management model, verifying its security and comparing it with other ones in terms of the execution time and key storage.

### 1.3. Motivation and Contributions

This paper focuses on the problem of reducing the number of secret keys stored at the KDC and proposes a method to reduce the number of secret keys stored at the KDC to only one key, regardless of the number of users involved. In this method, instead of keeping a shared secret key with every user involved in the network, the KDC stores a public stuff data related to every user and the user's secret key will be generated by taking the stuff data related to corresponding user, adding the lifetime of the secret key, and then hashing the resulting string using the manager secret key. The hash value will be used as the user's secret key. Therefore, the KDC will only store one key called the manager secret key. Furthermore, this paper combines the proposed method with our previous work [22], to build an efficient key management model, verify its security using a formal verification tool and compare it with other ones in terms of the execution time and key storage.

### 1.4. Organization of the Paper

This paper is organized into six sections, Introduction is in Section 1, Related Work are in Section 2, The Proposed Key Management Model is in Section 3, Security Verification is in Section 4, Performance Evaluation is in Section 5, and Conclusion is in Section 6.

## 2. RELATED WORK

This section gives a brief description of previous work presented in the literature on Key management and distribution.

M. Khalifa[7], proposed an enhanced authentication model based on Kerberos system. This model consists of two stages and is intended to provide protection for Kerberos from replay, screen shot, key logger, and password-guessing attacks. In the first stage, author used RSA encryption to secure the password saved in external device in order to avoid

transmitting password as clear text over the network. In the second stage, author used CRC algorithm as a complement to the first stage.

S. Arora and M. Hussain[8], proposed a key agreement protocol for generation and sharing a session key between two parties using symmetric key cryptography. In this protocol, a trusted third party is used only to establish the communication between the communicating parties. The session key is generated by the sender and receiver based on a cyclic group and its security is based on discrete logarithm problem. Authors analyzed the security of their protocol by using a formal verification tool and showed that it is safe and secure. However, due to the session key generation is done at the sender and receiver, this protocol reduces the computation overhead at the KDC. But it suffers from key storage overhead because it assumes that the KDC maintains a shared secret key with each party.

M. D. Nath and S. Karforma [11], Presented a model based on Kerberos, combined with IDEA algorithm. The authors applied the proposed model for logging into the electronic banking system, and showed that their model would be beneficial in nature, both to the bank and to the customer during electronic banking transactions, and also provide privacy, integrity and confidentiality in such environments.

Tbatou et al. [12], presented an authentication protocol for distributed systems based on Kerberos V5 and Diffie–Hellman models. The protocol contains three phases, registration phase, communication phase and renewal phase. The objective of this protocol was to protect passwords chosen by users for removing the dictionary attack and brute force attack from Kerberos. Authors demonstrated that their model provides a secure channel to a more secure password exchange.

J. Ghosh Dastidar [19], proposed a simplified form of the Kerberos system. This model depends on symmetric key cryptography and utilizes nonce and time-stamp to prevent replay attack. In this model, rather than using two separate entities, a Authentication Server (AS) and a Ticket Granting Server (TGS), as Key Distribution Center (KDC) as in Kerberos, it uses just a single entity that plays the role of both. In this model, the KDC stores  $N$  secret keys for all users involved.

A. Jesudoss and N. Subramaniam.[20], presented an enhanced version of Kerberos system to protect Kerberos from password-guessing attack and replay attack. This model uses additional secret key and nonce value for the initial authentication in Kerberos. In this model, the KDC stores two secret keys for every user in the network. This means, the KDC stores  $2N$  secret keys for all users, where  $N$  is the total number of users in the network.

**RESEARCH ARTICLE**

H. Saputra and Z. Zhao [21], presented a model for key management in the SCADA systems. Authors focused on the long-term keys management and proposed a method to update and refresh these keys dynamically. They showed that their method provides a flexibility to update the long-term keys and reduces the number of long-term keys stored at each endpoint.

Dua et al.[23], presented a modified version of Kerberos to avoid replay and password guessing attacks. In this system, the client sends three passwords to the KDC across the network, these passwords are used for authentication and key derivation. Due to sending the passwords over the network, if weak passwords are chosen by the client, this system may be exposed to attackers[20]. Moreover, this system suffers from key storage overhead at the KDC, due to, the KDC stores three secret keys for every user in the network.

P. Shalini and M. Kushwaha[24], presented an authentication and key distribution protocol. In this protocol, the session key is generated by the Trusted Server which distributes it between two communicating parties using symmetric key encryption. The attack on this protocol is demonstrated in our previous work[22].

**3. PROPOSED KEY MANAGEMENT MODEL**

In this section, we present the proposed model for reducing the number of secret keys stored at the KDC, known here as Key Managing Center (KMC). The notations used to describe the proposed model are listed in Table 1, and the following subsections describe the key hierarchy, proposed method sequence, and details of the proposed method.

Notation	Description
IDA, IDb	The identity of users A and B
KMC	Key Managing Center
[M]K	Encryption of message M with K
SKa	The secret key of user A
PKa/RKa	The public/private key pair of user A
Na, Nb	Nonce value chosen by users A and B
SDa	The stuff data related to the user A
L	The life time of the secret key
H	Hash function
SRNG	Pseudorandom Number Generator
	Concatenation of sets of strings

Table 1 Notations and their Description

**3.1. Key Hierarchy**

In the proposed model, in order to simplify key management and achieve the main objectives of this paper, the keys have been planned from the highest to the lowest into three levels: at the highest level are the public/private key pair and the manager secret key, at the middle level is the user secret key, and at the lowest level is the session key. The block diagram of Figure 1 shows the hierarchy of these keys.

**1. The public/private key pair**

The public/private key pair is used to protect the user's secret key. It has a very long life, usually several years.

**2. The manager secret key**

The manager secret key is the key used by the system manager to generate the user's secret key. It also has a very long life, usually several years.

**3. The user secret key**

The user secret key is used to encrypt and decrypt the messages that exchange session keys. Its lifetime is long, usually several months.

**4. The session key**

The key used to encrypt and decrypt the original data is called the session key. The session key exists only when two users wish to exchange data in a single communication session. Its lifetime is very short, usually only several minutes.

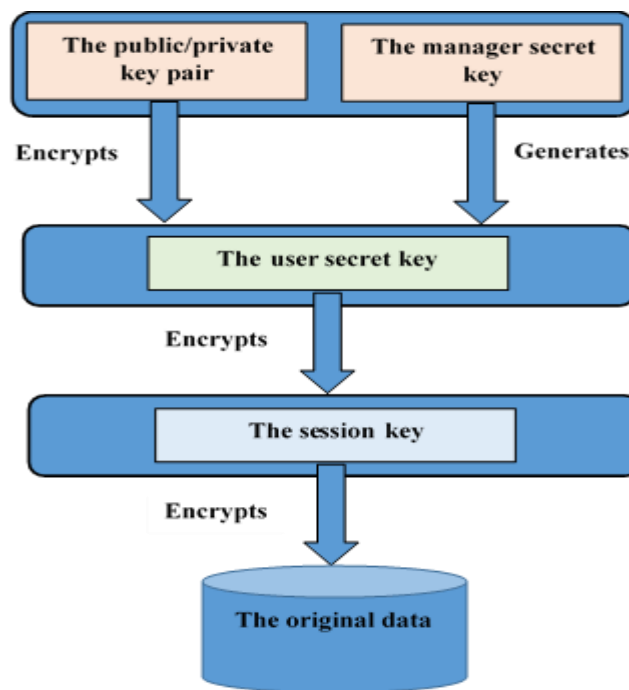


Figure 1 Block Diagram of Key Hierarchy Plan



**RESEARCH ARTICLE**

3.2. Proposed Method Sequence

In this subsection, we will provide the sequence of steps to be followed to design the proposed model and details of these steps will be provided in the next subsection. These steps are as follows:

1. All users in the network, including KMC have to generate their own public/private key pair.
2. Each user must register with the manager's database and obtain the KMC's Public Key. The user will send some public information about him/herself like: ID, Public Key, Phone Number, IP address, and so on. Then, the KMC generates a unique stuff data based on user identification and stores it along with the user's information in the manager's database.
3. To compute the user's secret key, the KMC uses a suitable hash function along with three parameters: the first one is the manager secret key, the second is the user stuff data stored in the manager database, and the third is the life time of the secret key. The output of the hash function is used as the user's secret key. Then, the KMC encrypts the user's secret key using the user's public key and sends it to the user.
4. The intended user receives the encrypted message, decrypts it using his/her private key and then stores the retrieved secret key in his/her memory device.
5. To reduce stored secret keys, the user may encrypt his/her secret key using his/her public key and retrieve it as needed.
6. If two users, say user A and B, wish to communicate with each other securely. User A or B can ask the KMC to generate a suitable session key for them. The KMC can then send this key to those users encrypted under the corresponding secret keys.

By those steps, there is no need to store more than one key securely, and the KMC should periodically change the secret key and the stuff data for all users based on the life time of the secret key.

3.3. Details of The Proposed Method

The proposed method includes three phases:

- Registration phase,
- Secret key establishment phase, and
- Session key distribution phase.

Figure 2, summarize the phases of the proposed model and the details of these phases will be described in the following sub-sections.

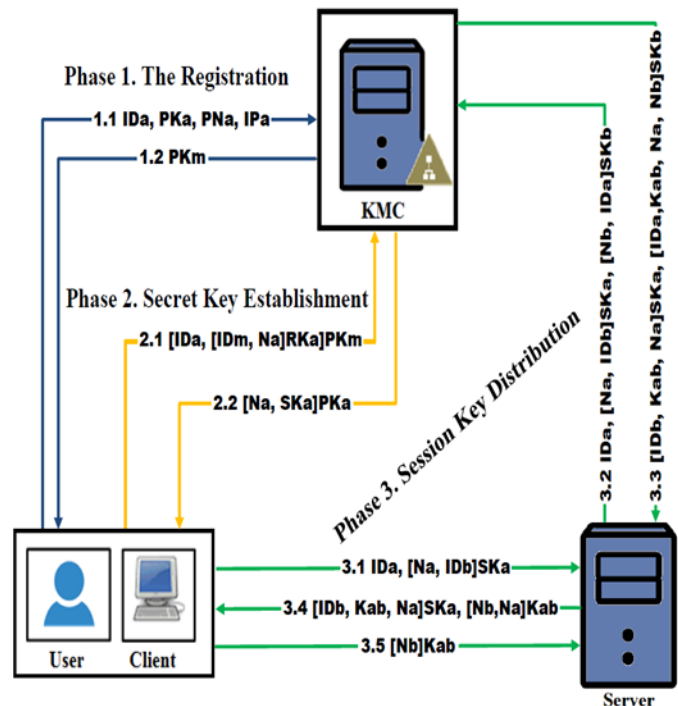


Figure 2 Block Diagram of the Proposed Model

3.3.1. Registration Phase

All users must first generate their own public/private key pair, register with the manager's database and obtain the KMC's Public Key. In this phase, each user sends the public information about him/herself like: ID, PK, PN, IP, and so on to the KMC. Then, the KMC receives the data sent by the user and verifies the user ID. If it exists in the manager database, the KMC replay back an error message. Otherwise, it generates a unique stuff data for the user based on user identification using a pseudorandom number generator according to equation (1).

$$SD = SRNG(ID \parallel PK \parallel PN \parallel IP) \tag{1}$$

Finally, the KMC Stores the stuff data along with the user information in its database and sends its public key to the user. To avoid the problem that arises if a public key is used by a party who is not the actual owner of the public key [5, 25]. This phase should be performed through a secure channel such as e-mail.

The algorithm 1 provides the pseudocode for generating the user stuff data and the block diagram of the registration phase is provided in Figure 3.

Stuff-Data-Generation-Algorithm(ID,PK,PN,IP)

Begin

**RESEARCH ARTICLE**

1. If ID exists then
  - 1.1 Return error-message
2. Else
  - 2.1  $SD = SRNG(ID \parallel PK \parallel PN \parallel IP)$
  - 2.2 Store SD along with ID, PK, PN and IP in the manager database
  - 2.3 Return SD
3. End If

End

Algorithm 1 Pseudocode for Generating the User Stuff Data

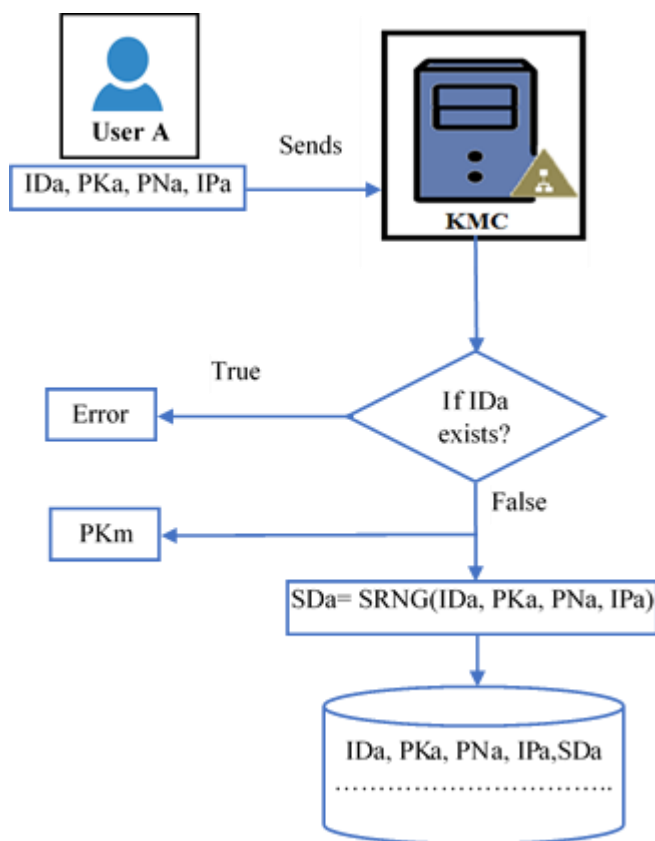


Figure 3 Block Diagram of the Registration Phase

3.3.2. Secret Key Establishment Phase

This phase describes the secret key establishment process between a user and the KMC. The KMC will compute a secret key for every user in the network at the request of the user. The following steps describe the protocol used in this phase:

1. User A sends a generation request to the KMC. This request includes the identity of the user (IDa) along with the identity of Key Managing Center (IDm) and nonce

value (Na) signed with the user's private key RKa, all collected together and encrypted by the KMC's public key PKm.

$$A \rightarrow KMC : [IDa, [IDm, Na]RKa]PKm$$

2. KMC receives the request and decrypts it using its private key. Afterwards, it uses IDa to search of PKa in its database. Then it verifies the user's signature:

- If true, the KMC generates the user's secret key by applying a hashing algorithm along with three parameters, the first one is the manager secret key, the second is the user's stuff data stored in the manager database, and the third is the life time of the user's secret key. The equation 2 provides the formulation of hashing function for generating the user's secret key.

$$SKa = H(SKm \parallel SDa \parallel L). \tag{2}$$

The block diagram of Figure 4. summarizes our proposed method to generate the user's secret key and the algorithm 2. provides the pseudocode for generating the user secret key.

- Otherwise, the KMC reject this request and terminate the connection with the user.
3. KMC constructs a message for User A encrypted under the user's public key. This message includes the user secret key and Nonce Na. After that it sends this message to that user.

$$KMC \rightarrow A : [Na, SKa]PKa$$

4. User A receives the encrypted message, decrypts it with his/her private key, and then verifies the integrity of Nonce Na. So, User A would be sure this message is fresh and its originator is KMC and not someone else.

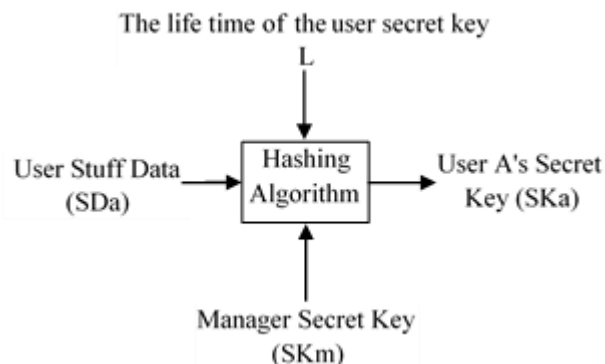


Figure 4 Block Diagram of the Secret Key Generation

Secret-Key-Generation-Algorithm(ID)

Begin

1. L = get the life time of secret key

**RESEARCH ARTICLE**

2. SK<sub>m</sub> = get the manager secret key
3. SD = retrieve the user stuff data from the manager database based on ID
4. PK = retrieve the user public key from the manager database based on ID
5. V = verify the user's signature by using PK
6. If V is true then
  - 6.1 SK = hash(SK<sub>m</sub> || SD || L)
  - 6.2 Return SK
7. Else
  - 7.1 Return error-message
8. End If

End

Algorithm 2 Pseudocode for Generating the User Secret Key

3.3.3. Session Key Distribution Phase

This phase describes the distribution of session key between two users with the help of the KMC, if any two users want to communicate with each other securely, they can obtain a session key from the KMC which in turn generates a session key, encrypts it using the secret keys that it shares with those users and sends it to them.

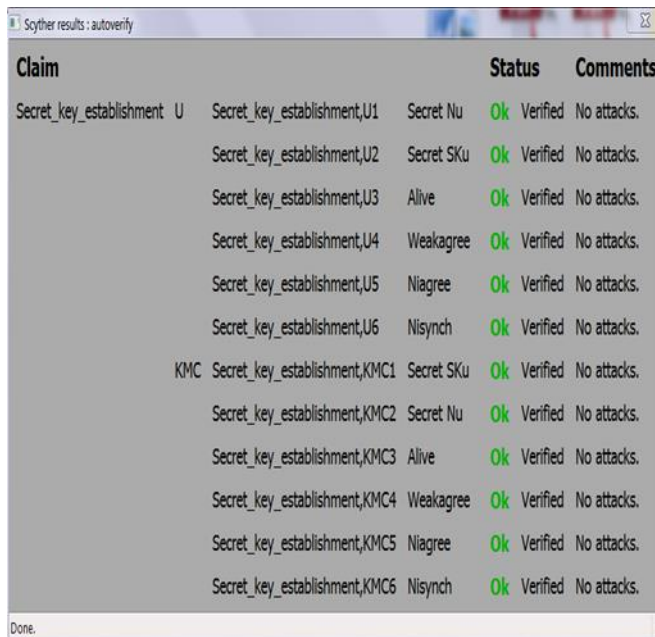
Here, we have used an improved key distribution protocol presented in our previous work[22]. That protocol has been proposed based on the protocol proposed in[24]. It uses symmetric key cryptography for distribution of session key between two parties with the help of a trusted third party. Following are the steps of the protocol and more details about this protocol are given in[22].

1. A → B: ID<sub>a</sub>, [Na, ID<sub>b</sub>]SK<sub>a</sub>
2. B → S: ID<sub>a</sub>, [Na, ID<sub>b</sub>]SK<sub>a</sub>, [Nb, ID<sub>a</sub>]SK<sub>b</sub>
3. S → B: [ID<sub>b</sub>, Kab, Na]SK<sub>a</sub>, [ID<sub>a</sub>, Kab, Na, Nb]SK<sub>b</sub>
4. B → A: [ID<sub>b</sub>, Kab, Na]SK<sub>a</sub>, [Nb,Na]Kab
5. A → B: [Nb]Kab

4. SECURITY VERIFICATION

The proposed model has been verified using the formal verification tool called Scyther. Scyther is an automatic verification tool and widely used and accepted for the verification of security protocols [26], against various types of attacks[27]. Scyther tool has many advantages like: the verification of protocols with unbounded number of sessions, it supports multi-protocol analysis[28] and provides a property to generate claims automatically. Scyther takes as

input a role-based description of a security protocol and the security goals are specified using claim events. The language used to write security protocols in Scyther is Security Protocol Description Language (SPDL)[28-30]. Here, the secret key establishment protocol was implemented in Scyther and verified with auto-verification property. Table 2 shows the verification results of this protocol.



Claim	Status	Comments
Secret_key_establishment,U	Secret Nu	Ok Verified No attacks.
Secret_key_establishment,U1	Secret SKU	Ok Verified No attacks.
Secret_key_establishment,U2	Alive	Ok Verified No attacks.
Secret_key_establishment,U3	Weakagree	Ok Verified No attacks.
Secret_key_establishment,U4	Niagree	Ok Verified No attacks.
Secret_key_establishment,U5	Nisynch	Ok Verified No attacks.
Secret_key_establishment,U6	Secret SKU	Ok Verified No attacks.
KMC Secret_key_establishment,KMC1	Secret Nu	Ok Verified No attacks.
Secret_key_establishment,KMC2	Secret SKU	Ok Verified No attacks.
Secret_key_establishment,KMC3	Alive	Ok Verified No attacks.
Secret_key_establishment,KMC4	Weakagree	Ok Verified No attacks.
Secret_key_establishment,KMC5	Niagree	Ok Verified No attacks.
Secret_key_establishment,KMC6	Nisynch	Ok Verified No attacks.

Table 2 Verification Results of Secret Key Establishment Protocol

From the verification results shown in Table 2 above, we can observe that the protocol fulfill all security claims. This means the protocol completely fulfill the security goals and hence no potential attacks can be performed.

5. PERFORMANCE EVALUATION

In order to demonstrate that the proposed key management model is more efficient, this section presents the simulation environment and the evaluation results of the proposed model.

5.1. Simulation Environment

The simulation has been performed using Sockets in C# language 2012 on Windows 7 64-bits, Core i5-2520M CPU and 4GB RAM. In this simulation, we consider a network that consists of a client, service server and KMC. In our implementation, we have used the libraries provided in .NET framework, where RSA algorithm has been selected as the public-key encryption algorithm with 2048-bits key to protect the secret keys between a user and the KMC, AES of 256-bits for symmetric encryption to distribute the session key between any two users with the help from the KMC and to encrypt the original data exchanged on the network, and

**RESEARCH ARTICLE**

SHA-256 for hash function to generate the required symmetric keys.

5.2. Results and Discussion

Here, we evaluate the performance of the designed model and compare it with the two enhanced versions of the widely used Kerberos system[19, 20] in terms of the execution time and key storage.

5.2.1 Execution Time Evaluation and Comparison

As we mentioned in Section 3 above, the proposed model in this paper includes three stages: the first stage does not include any cryptographic process and is a one-time operation, so this stage will not be included in this evaluation.

The second stage includes two cryptographic operations: public-key encryption/decryption and hash function. Table 3 shows the execution time of the cryptographic operations performed in this stage.

Operation	The Execution Time (in MS)
RSA-2048 encryption	3
RSA-2048 decryption	57
SHA-256	4

Table 3 Execution Time of the Secret Key Establishment

From the Table 3, we can find that the RSA consumes the more time. This is because of the many computations with very large numbers involved in performing RSA, especially decryption is performing with computing of a large number to a huge power. However, the performance of this stage does not affect the system performance in general since this stage is used only occasionally to update the secret key between a user (client/server) and the KMC and this secret key is valid for a long period of time.

The last stage aims to establish the encryption key (the session key) between the communicating parties with the help of the KMC. When a user needs to communicate with the service server, she/he needs a session key. This session key is usually valid for a short period of time. Because of this, this stage occurs frequently and consequently this stage affects the system performance. So we have taken into account the execution time consumed by accomplishing this stage to compare the proposed protocol with the other protocols.

The selected protocols were implemented according to [19, 20] and the proposed protocol according to the session key distribution protocol described in subsection 3.3.3. For each protocol, the time consumed by performing session key distribution 50 times has been calculated at each entity

(Client, Server and KDC), and the simulation results are depicted in Figures. 5-7.

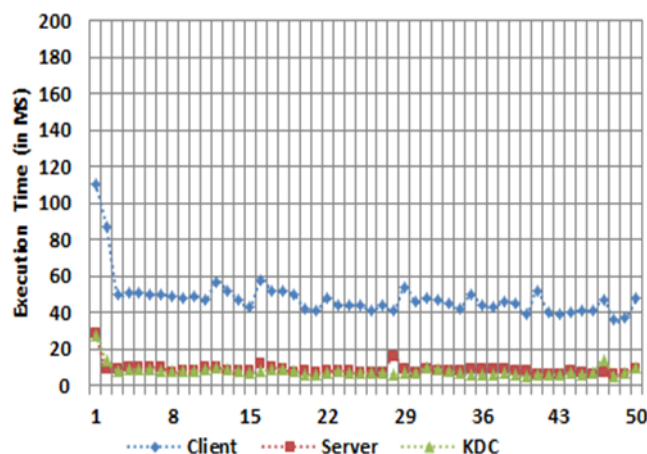


Figure 5 Execution Time of the Proposed Protocol

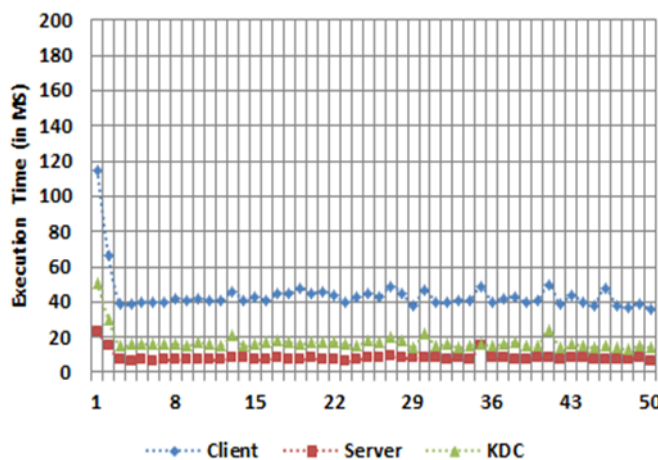


Figure 6 Execution Time of the Protocol in [19]

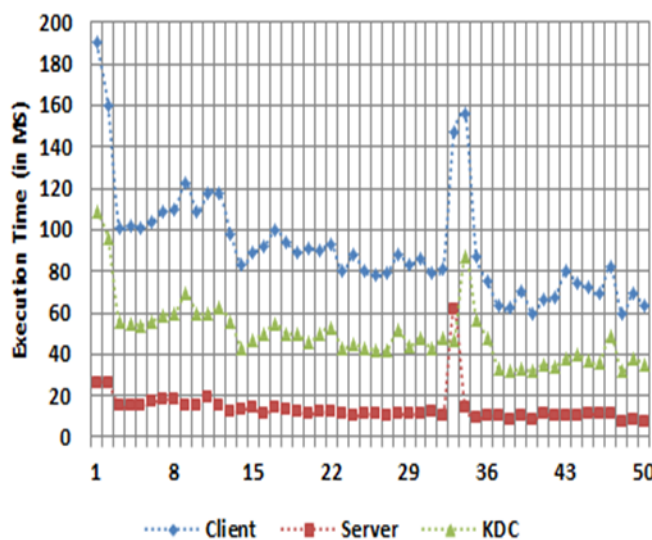


Figure 7 Execution Time of the Protocol in [20]

**RESEARCH ARTICLE**

In addition, we have calculated the average of the execution time consuming for all executions of each protocol up to 50 times. Figure 8 shows the comparison of average execution time for all protocols. The results shown in these figures demonstrate that the execution time consumed by the proposed model is shortest, while the execution time consumed by the others is longest. As we can find that the protocol proposed in [20] is the most execution time consuming protocol due to it requires additional encryption operations for the initial authentication between a user and the KDC.

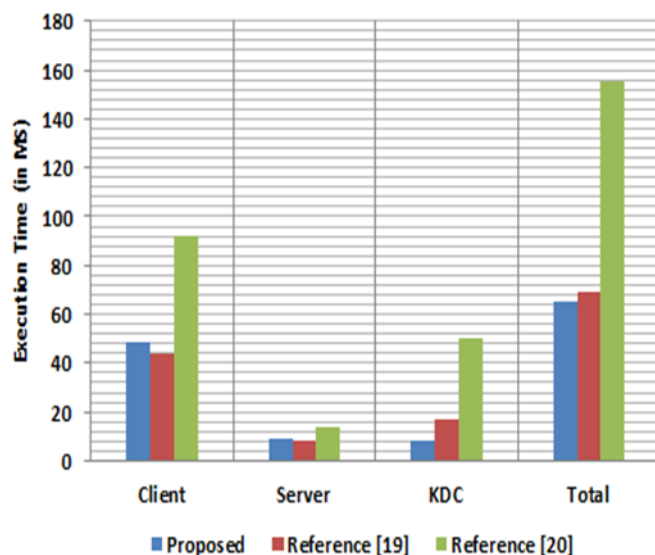


Figure 8 Average Execution Time Comparison

5.2.2. Key Storage Evaluation and Comparison

The storage requirement takes two forms: storage at the users and storage at the KDC. In the proposed model, there are several advantages in using the key hierarchy plan. One of these advantages is to reduce the number of key-encrypting keys which have to be stored securely. However, in our model, each user has two secret keys (key-encrypting keys): the private key and the secret key. In order to reduce the number of secret keys stored at the user, the user may encrypt his/her secret key using his/her public key. Thus, each user only needs to store one key securely. The same scenario goes for Key Managing Center.

The main advantage of the proposed key management model is that, the Key Managing Center (KMC) does not have to store the secret keys of all users involved in the network. Instead of storing N secret keys at the KMC for all users, the KMC stores a unique stuff data related to each user. This stuff data is not secure information and the user's secret key is generated based on this stuff data as described in Section 3. The comparison between the proposed model and the two enhanced versions of the widely used Kerberos system in

term of number of keys stored at the KDC is shown in Figure 9. In this comparison it is easy to note that the KDC in the proposed model only needs to store one key securely, unlike the other models in which the KDC stores N secret keys as in the protocol proposed in [19] and stores 2N secret keys as in the protocol proposed in [20].

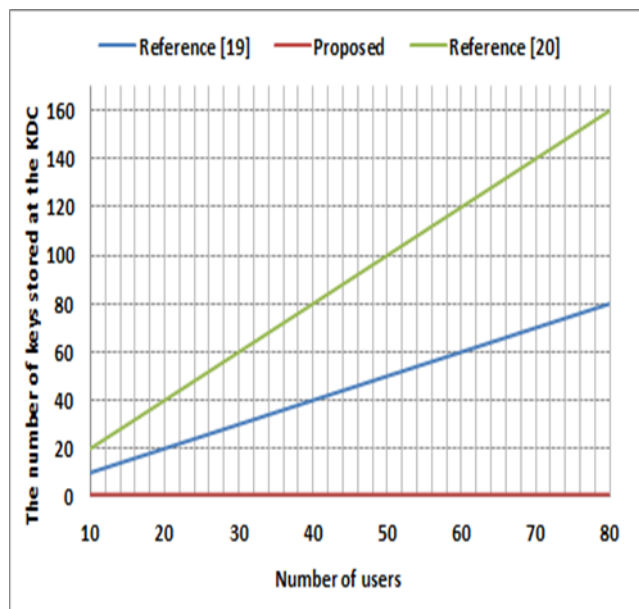


Figure 9 Performance Comparison of Key Storage

6. CONCLUSION

In this paper, we have focused on the problem of reducing the number of secret keys stored at the KDC and have proposed a method to reduce the number of secret keys stored at the KDC to only one key. Furthermore, we have combined the proposed method with our previous work and have built an efficient key management model. The main advantage of this model is that, the Key Managing Center (KMC) does not have to store N secret keys (key encryption keys) in its database. Instead of that, it stores a unique stuff data related to each user involved. This stuff data is not secure information. Therefore, the KMC only needs to store one key securely called the manager secret key and use it to generate the secret keys for users involved. In addition, each user involved in the network only needs to store one key securely. This will extremely reduce the key storage space and make our model suitable for a large-scale network. The performance of our model has been evaluated and the protocols used in this model have been verified by a formal verification tool called Scyther, the evaluation and verification results showed that the model is efficient and secure.

REFERENCES

[1] D. P. Sumalatha and D. C. K. Priya, "A Prototype Implementation for Public Key Infrastructure Based on Transport Layer Security,"



**RESEARCH ARTICLE**

International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 6, pp. 83-88, September - October 2017.

[2] R. Abobeah, M. Ezz, and H. Harb, "Public-key cryptography techniques evaluation," International Journal of Computer Networks and Applications, vol. 2, pp. 64-75, 2015.

[3] S. Sharma, "Cryptography: An art of writing a secret code," International Journal of Computer Science & Technology, vol. 8, pp. 26-30, 2017.

[4] K. H. K. Alibraheemi, "Robust Biometrics-Based Authentication Scheme for Cryptographic Keys Distribution," International Journal of Applied Engineering Research, vol. 13, pp. 1415-1420, 2018.

[5] W. Stallings, Cryptography and Network Security: Principles and Practice: Pearson Prentice Hall, 2017.

[6] Jincy Sebastian and S. Jose, "Implementation of Two-Server Password-Based Authentication," International Journal of Innovative Research in Computer and Communication Engineering, vol. 3, pp. 11608-11614, 2015.

[7] M. Khalifa, "Enhanced Kerberos Authentication For Distributed Environment Using Two Phases Security," An international journal of advanced computer technology, vol. 6, pp. 2323-2329, 2017.

[8] S. Arora and M. Hussain, "Secure session key sharing using symmetric key cryptography," in 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp. 850-855.

[9] S. Budiyo, G. B. Santosa, and F. R. I. Mariati, "Upgrading the S-NCI Key Establishment Protocol Scheme to be Secure and Applicable," in IOP Conference Series: Materials Science and Engineering, 2018, p. 012002.

[10] M. Narendra and S. S. Raja, "A Study of Using Authenticated Key Exchange Protocols to Boost The Efficiency of Parallel Network File System," Journal of Critical Reviews, vol. 7, pp. 1883-1890, 2020.

[11] M. D. Nath and S. Karforma, "Object-Oriented Modelling Of Kerberos Based Authentication Process In E-Banking Transaction," International Journal of Computer Sciences and Engineering, vol. 6, pp. 1-5, 2018.

[12] Z. Tbatou, A. Asimi, Y. Asimi, Y. Sadqi, and A. Guezzaz, "A New Mutual Kerberos Authentication Protocol for Distributed Systems," Int. J. Netw. Secur., vol. 19, pp. 889-898, 2017.

[13] L. Wu, J. Fan, Y. Xie, and J. Wang, "An improved authentication and key agreement scheme for session initial protocol," KSII Transactions on Internet and Information Systems (TIIS), vol. 11, pp. 4025-4042, 2017.

[14] K. Bakare, S. Junaidu, and M. Ahmed, "Improved Secure Biometric Authentication Protocol," International Journal of Applied Information Systems, vol. 12, pp. 49-56, 2020.

[15] T. A. Khaleel, "Analysis and Implementation of Kerberos Protocol in Hybrid Cloud Computing Environments," Engineering and Technology Journal, vol. 39, pp. 41-52, 2021.

[16] Z. Tbatou, A. Asimi, C. E. Balmany, and Y. Asimi, "A Novel Architecture of a Strong and Mutual Authentication Protocol for Distributed Systems," Engineering Letters, vol. 28, 2020.

[17] P. Bhadle, S. Gugale, S. Trar, H. Kaur, and S. Salve, "Kerberos Authentication System using Public key Encryption," International Journal of Computer Science and Information Technologies, vol. 5, pp. 1930-1933, 2014.

[18] J. Sun and Z. Gao, "Improved mobile application security mechanism based on Kerberos," in Proceedings of 2019 4th international workshop on materials engineering and computer sciences, 2019, pp. 108-112.

[19] J. G. Dastidar, "An Authentication Protocol based on Kerberos," Journal of Engineering Research and Application, vol. 7, pp. 70-74, 2017.

[20] A. Jesudoss and N. Subramaniam, "Enhanced Kerberos authentication for distributed environment," Journal of Theoretical and Applied Information Technology (JTAIT), vol. 69, pp. 368-374, 2014.

[21] H. Saputra and Z. Zhao, "Long term key management architecture for SCADA systems," in 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), 2018, pp. 314-319.

[22] A. Yasser, Alahmadi and N. Saleh, Allassali, "An Improved Key Distribution Protocol Using Symmetric Key Cryptography," International Journal of Computer Sciences and Engineering (IJCSE), vol. 8, pp. 21-26, 2020.

[23] G. Dua, N. Gautam, D. Sharma, and A. Arora, "Replay attack prevention in Kerberos authentication protocol using triple password," International Journal of Computer Networks & Communications (IJCNC), vol. 2, pp. 59-70, 2013.

[24] P. Shalini and M. Kushwaha, "Mutual Authentication and Secure Key Distribution in Distributed Computing Environment," International Journal of Advanced Research in Engineering and Technology (IJARET), vol. 11, pp. 378-390, 2020.

[25] V. Lozopone, "Analyze encryption and public key infrastructure (PKI)," International Journal of Information Management, vol. 38, pp. 42-44, 2018.

[26] R. Amin, P. Lohani, M. Ekka, S. Chourasia, and S. Vollala, "An enhanced anonymity resilience security protocol for vehicular ad-hoc network with Scyther simulation," Computers & Electrical Engineering, vol. 82, p. 106554, 2020.

[27] M. Saffkhani, N. Bagheri, and M. Shariat, "On the security of rotation operation based ultra-lightweight authentication protocols for RFID systems," Future Internet, vol. 10, pp. 1-15, 2018.

[28] M. H. Alzuwaini and A. A. Yassin, "An Efficient Mechanism to Prevent the Phishing Attacks," Iraqi Journal for Electrical & Electronic Engineering, vol. 17, 2021.

[29] S. Bojjagani, D. D. Brabin, and P. V. Rao, "PhishPreventer: a secure authentication protocol for prevention of phishing attacks in mobile environment with formal verification," Procedia Computer Science, vol. 171, pp. 1110-1119, 2020.

[30] E. Munivel and A. Kannammal, "New authentication scheme to secure against the phishing attack in the mobile cloud computing," Security and Communication Networks, vol. 2019, pp. 1-11, 2019.

Authors



**Yasser Ali Alahmadi**, pursued B.Sc. CS from Sana'a University, Yemen. He is currently pursuing Master of Computer Science, Department of Computer Science, Sheba Region University, Yemen. His interest research area: Information Security and C# Programming.



**Dr. Mokhtar Alsorori**, pursued B.Sc. CS from Al-Neelain University, Sudan in 2003, M.Sc. IT from Mysore University, India in 2007 and Ph.D. in Computer Science from Kakatiya University, India in 2020. He is currently working as Assistant Professor in Department of Information Systems, Sheba Region University, Yemen. His interest research area: Cyber Security and Digital Image Watermarking.



**Dr. Saleh Noman Abdullah Allassali**, pursued B.Sc. CE from KSU University, Saudi Arabia in 1988, M.Sc. CS from Pune University, India in 2000 and Ph.D. in Information Security from SRTM University, India in 2005. He is currently working as Associated Professor in each of Department of Computer Sciences, Sheba Region University, Science and Technology University, Yemen. He has published more than 8 research papers in reputed international journals. His main

research work focuses on Cryptography Algorithms and Random Number Generators.



**RESEARCH ARTICLE**

**How to cite this article:**

Yasser Ali Alahmadi, Mokhtar AlSORori, Saleh Noman Alassali, “Reduce the Memory Used in Key Management for Security Systems”, International Journal of Computer Networks and Applications (IJCNA), 8(4), PP: 412-421, 2021, DOI: 10.22247/ijcna/2021/209707.