Research Article

Saman Shojae Chaeikar*, Ali Ahmadi, Sasan Karamizadeh, and Nakisa Shoja Chaeikar

SIKM – a smart cryptographic key management framework

https://doi.org/10.1515/comp-2020-0167
received April 28, 2020; accepted January 18, 2021

Abstract: For a secure data transmission in symmetric cryptography, data are encrypted and decrypted using an identical key. The process of creating, distributing, storing, deploying, and finally revoking the symmetric keys is called key management. Many key management schemes are devised that each one is suitable for a specific range of applications. However, these schemes have some common drawbacks like the hardness of key generation and distribution, key storage, attacks, and traffic load. In this article, a key management framework is proposed, which is attack resistant and transforms the current customary key management workflow to enhance security and reduce weaknesses. The main features of the proposed framework are eliminating key storage, smart attack resistant feature, reducing multiple-times key distribution to just one-time interpreter distribution, and having short key intervals – minutely, hourly, and daily. Moreover, the key revocation process happens automatically and with no revocation call.

Keywords: cryptographic key management, symmetric key, secure communications, key exchange, key refreshment

1 Introduction

A correct key management practice results in having a reliable use of cryptography, and accordingly, good information management [1]. Cryptography falls into two types: symmetric and asymmetric. Symmetric class is when both parties share an identical secret – a key – for encrypting and decrypting their information. To generate, distribute, and finally revoke a key, asymmetric key management framework is required. Several practices are introduced for key generation or key issuance, such as random number generator and key derivation functions. An important factor in key generation process is key strength which analyzes how likely a key is to be broken by cryptanalysis attacks. To enhance the key strength, the weak keys that are normally influenced by the plain text patterns must be omitted from the key pool [2]. Long cryptographic keys deliver higher security, and with today's equipment, cracking a 256-bits key or longer is impossible computationally. The longer keys normally provide higher security, but increase the computational costs as well [2].

After generating a key, it must be distributed between the engaged parties [3]. Sending keys in a clear text format results in compromising the keys. Therefore, the process of key distribution needs to be protected against adversaries. The first-ever key distribution scheme is the diplomatic bag [4]. In 1975, Diffie–Hellman introduced a blind channel establishment technique, which is still one of the best key distribution practices [5]. Later, the German Army Enigma (GAE) offered to combine a privately distributed key schedule and a piece of user-chosen secret key for every message [6]. Pretty Good Privacy (PGP) is another method that encrypts a symmetric key by means of an asymmetric key [7]. Another practice is key wrapping: encapsulating or hiding a key within another pre-distributed key [7]. When the keys should be related together or a periodical secure exchange is required, a proper choice is using a master key and deriving subsidiary keys when they are required. Another common method is the key predistribution [8,9].

Final key management step – key revocation – is informing all nodes that the current key is no more usable and a new key will be given for future sessions [10]. This may happen by issuing a revocation call, or through a predefined key lifetime.

Key storage – storing the keys for future use – is a critical key management issue that increases the probability

* **Corresponding author: Saman Shojae Chaeikar,** Department of Information Security, Faculty of Computer Engineering, Iranians University an e-Institute of Higher Education, Tehran, Iran, e-mail: saman.shoja@iranian.ac.ir
Ali Ahmadi: Department of Artificial Intelligence, Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran, e-mail: ahmadi@kntu.ac.ir
Sasan Karamizadeh: Department of ICT Security, ICT Research Institute, Tehran, Iran, e-mail: s.karamizadeh@itrc.ac.ir
Nakisa Shoja Chaeikar: Department of Information Security, Faculty of Computer Engineering, Iranians University an e-Institute of Higher Education, Tehran, Iran, e-mail: nks.shoja@gmail.com

of compromising the keys [11,12]. Storing them in a raw bit-string format ignores the needed preliminary security countermeasures, and accordingly, various techniques are devised to overcome this problem. A common method is encrypting keys by an application and decrypting after entering a password [13,14].

Key length and frequency of key replacement [15] are two very important key management considerations. Employing long keys and replacing them periodically enhances security. When a long key is chosen, attacks like exhaustive key search are not applicable or at least are highly time and resource consuming. The keys should ideally be unique for each transaction or message [16]; however, due to the high cost and hardness of the process, this is not applicable in all circumstances.

Cryptographic keys are either symmetric or asymmetric. Symmetric key cryptography is a class in which both parties use an identical or trivially related secret – keys are derived from another. In other words, the users employ an identical key for enciphering and deciphering processes. Symmetric keys also might be called the secret key, one key, shared key, single key, or private key in which private key and secret key might mistakenly be considered for similar terminologies in the public-key cryptography domain [17,18]. In terms of the method of data processing, symmetric algorithms are classified into two classes: stream cipher and block cipher. Stream cipher feeds the message bites one at a time into the encryption/decryption process, while block ciphers consider data as specific chunks – normally 64 or 128 bits – and encrypt/decrypt one chunk at each time [19,20]. The most common symmetric block cipher algorithms are AES, Blowfish, DES, Triple DES, Serpent, and Twofish [21,22]. Symmetric keys operate hundreds to thousands of times faster than asymmetric keys, as fewer computations are needed to perform the processes. The keys should be updated or replaced periodically to avoid being discovered by the adversaries [23,24].

From the key types perspective, the five main key management classes are network-wide key, pairwise key, random pairwise key, public key, and group key [24]. The network-wide key is the simplest scheme that predistributes a key between all nodes before deploying the network. Its main drawbacks are requiring key storage and network-wide key compromising, if one node compromises [25]. In the pairwise key, every node has a predistributed key in common with the other nodes. To deploy this method, for n nodes in a network, $n - 1$ pairwise keys are required. It provides a high level of security, while the size of the key storage and consumed memory increases and scalability drops [26]. The third solution is the notion of random pairwise key (RPK) that solves the key storage problem of the previous method by randomly choosing a set of pairwise keys from a key source [27]. By moving from symmetric cryptography to asymmetric, public-key cryptography appears as a solution to the counted problems. It produces strong keys with a high level of security with the cost of increasing the computational cost and requiring additional infrastructures – like certificate authority [24]. The final solution is a combination of network-wide key and pairwise key methods. Intergroup data exchange uses the pairwise key method, whereas the intragroup communications secure by a shared network-wide key. This combination has a higher level of scalability, is more resilient, and is resistant to node capture attacks compared to both methods. Its main drawback is in the process of forming and organizing node groups [28].

With regards to key distribution methodologies, the current practices are classified into three categories: key predistribution, postdeployment, and hybrid method [24]. In key predistribution, the keys are embedded within the nodes before deploying the network. Its most critical problem is the need for large-size key storage. Moreover, revoking the embedded keys and distributing new ones – rekeying – is a very costly and time-consuming process [29]. The other practice, postdeployment method, dynamically forms the keys in the nodes once the network is developed. Due to high resource consumption, rekeying is the weakness of this method, and accordingly, it is applicable when a limited number of keys are required [29]. Hybrid key distribution is a combination of key predistribution and postdeployment methods. Although it is very scalable, it requires the resources of the both aforementioned methods. Storing a limited number of predistributed keys results in more need for postdeployment keys – a very costly process – whereas embedding a large number of predistributed keys reduces the resilience of the network against the attacks [30].

The master key is a cryptographic key generation algorithm that derives the required keys from an initialization vector, some context, and a label. The key that the resultant keys are derived from is called the key derivation key. A key derivation function produces the derived keys by an automated key establishment process or a random bit generator [31]. The KDFs may require multiple iterations to produce the desired key length. The master key has three well-known KDF modes: counter, feedback, and double-pipeline iteration. In the first mode, the result of the pseudorandom function is an iteration value – a counter. The second mode uses the output of the previous iteration together with a counter as the input of the consequent iteration. Both counter and feedback modes

produce the keys through a single pipeline, while in the double-pipeline iteration mode, this has increased to two pipelines [31].

To address the aforementioned issues, this article introduces an attack resistant and Smart Interpretative Key Management (SIKM) framework – an enhanced version of our previous research [32] – that changes the main workload of key management from the server side to the client side. This change of the workflow reduces server and network traffic and increases security by means of producing fresh and time-dependent keys. The following sections describe the structure of the proposed SIKM framework, study the level of the reduced traffic, and analyze the SIKM's attack resistance feature.

2 Workflow and components

The workflow in SIKM initiates from a server that is responsible to generate a light-weight program that here is called the interpreter. Once the interpreter created, the server uploads its encrypted version to either a public or a dedicated FTP server. Then, all of the nodes download and decrypt the interpreter using the given key from the server. In the next step, all nodes unify their date and time with the defined time server. Upon completing these steps, the nodes can generate synchronized keys and hold secured sessions. Since the nodes produce fresh and dynamic keys at the client side, the necessity of storing keys is resolved, and always a fresh minutely, hourly, or daily key is ready in the hand of the nodes. SIKM identifies exhaustive key search and replay attack, and reduces the likelihood of man-in-the-middle attack. In light of having a constant key lifetime, the key revocation process has been eliminated and the keys revoke automatically. The SIKM framework is formed by two components: a server and an interpreter. The server has four main tasks as follows.

- Generating the interpreter.
- Distributing the interpreter between the authenticated nodes.
- Monitoring security status of nodes.
- Issuing the interpreter revocation call.

An interpreter is a light-weight piece of software that generates keys based on its intrinsic knowledge. The knowledge consists of time and date references (a time zone and calendar), a bit-stream source address, the interpreting method (or the key extraction algorithm), 36 embedded digits, and an interpreter revocation code.

After producing the interpreter, the server uploads its encrypted copy to either a dedicated or a publicly available source. The legitimate nodes receive the decryption key through a secure channel from the server. After decrypting and running the downloaded interpreter, it downloads a bit-stream from the given network address, and then synchronizes its time and date with the defined public source on the Internet or the server. Since the nodes utilize identical key initialization resources, they produce identical time-dependent keys synchronously. The key production time interval options are minutely, hourly, or daily, and the keys expire automatically after the defined period without any revocation call.

2.1 Server

In the SIKM key management framework, the first component is a server that produces an interpreter and manages the sessions until expiring. After generating the interpreter, the server uploads it to a dedicated or public space on the network. The authorized nodes receive the interpreter address and its decryption key from the server and then install it. Hereafter, the server's main tasks are managing new nodes, analyzing the nodes' security status, issuing the interpreter revocation call, and answering time synchronization requests.

The overall SIKM process is shown in Figure 1. In the current practices, the workflow mainly is from the server toward the nodes, while this flow is changed in SIKM, and only in very limited stages, the server is involved.

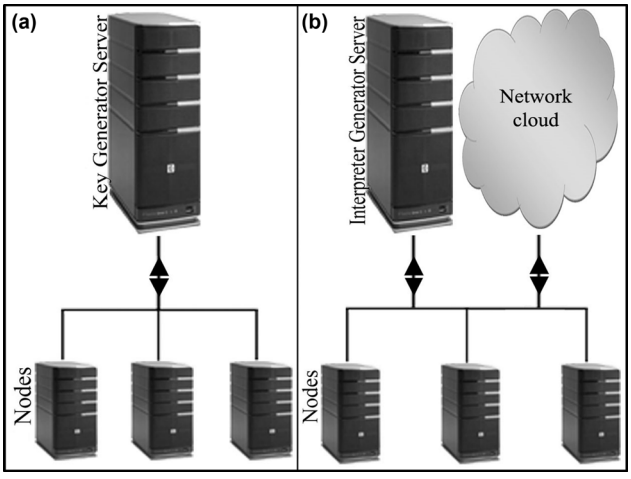

Figure 1: (a) The workflow in the common key management practices and (b) the workflow in SIKM.

2.2 Interpreter

The embedded items within the interpreter are a set of 36 digits, bit-stream source address, time and date (agreed time zone and calendar), an interpreter revocation code, bit-stream interpreting knowledge (a key generation algorithm), and in specific cases hardware specifications. The overall key generation process is illustrated in Figure 2.

Time and date: The nodes in SIKM may be distributed all over the world. Therefore, as time and date parameters are involved in key generation, these must be identical in all nodes to let them produce synchronized keys. To this end, all nodes first synchronize their local time and date with the server, or any other defined sources, and then convert these items into the defined calendar and time zone.

Thirty-six digits: As date and time are involved in key generation, these must be in a full format to construct a set of 12 digits – eight digits for date and four digits for time. Triple times concatenation of the 12 digits shapes a 36 digit number that is needed for the key extraction algorithm. To make it impossible for attackers to crack these 36 digits, the numbers are added to the embedded 36 digits with no carry in each step. The result is a new 36 digits that change at every key generation interval (Figure 3).

Bit-stream source: This an initialization factor that helps in synchronizing the key generation process between all nodes. It is extracted from the first n bits of the defined file or any data source on the network. To this end, all of the interpreters download the first required n bits and feed them into the key extraction algorithm. The bit-stream source update intervals should be known to the interpreter, as any change results in producing unsynchronized keys.

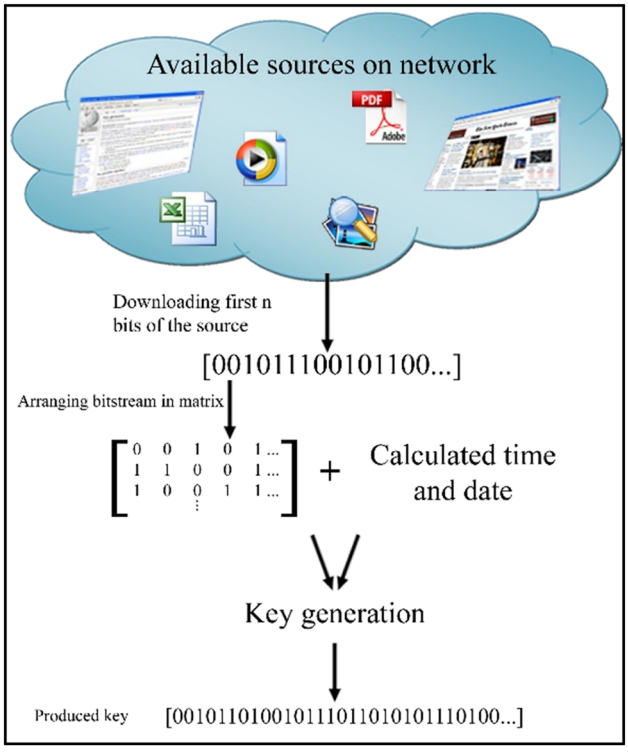

Figure 2: The overall process of key generation.

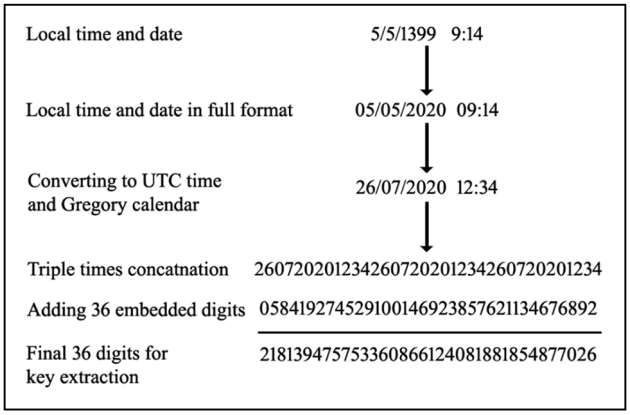

Figure 3: Calculating 36 digits based on the local date and time.

Key extraction algorithm: To produce a key, the downloaded bits are arranged in a matrix and then surveyed according to the final 36 digits. Starting from the index (0, 0), the algorithm selects the bits horizontally for odd numbers, and vertically for even numbers. This key extraction process continues until the key is fully produced. If a column or row ends before picking up the necessary number of bits, the process continues from the next column or row, respectively. By reaching the final item in a row, the process continues from the next row, and for each column from the next column. After the final row or column, the process continues from the first one. If the 36 digits end before producing the required key length, the key extraction process reuses the digits. Figure 4 visualizes the matrix key extraction technique.

Interpreter revocation code: This an embedded secret code within the interpreter that receiving it means the current version of the interpreter is expired due to the decision of the administrator, compromising the interpreter or a periodical replacement. To issue the call, the server encrypts the revocation code using the current key and then sends it to the nodes.

Hardware specifications: SIKM is applicable when the nodes are well known and trustable for a long term. In cases like a network of automated teller machines (ATMs), the specification of the used hardware in all nodes is known. Therefore, to enhance security, the interpreter could be configured to produce keys when the hardware specifications match the given specifications. This helps to avoid producing the keys if the interpreter is compromised and runs on

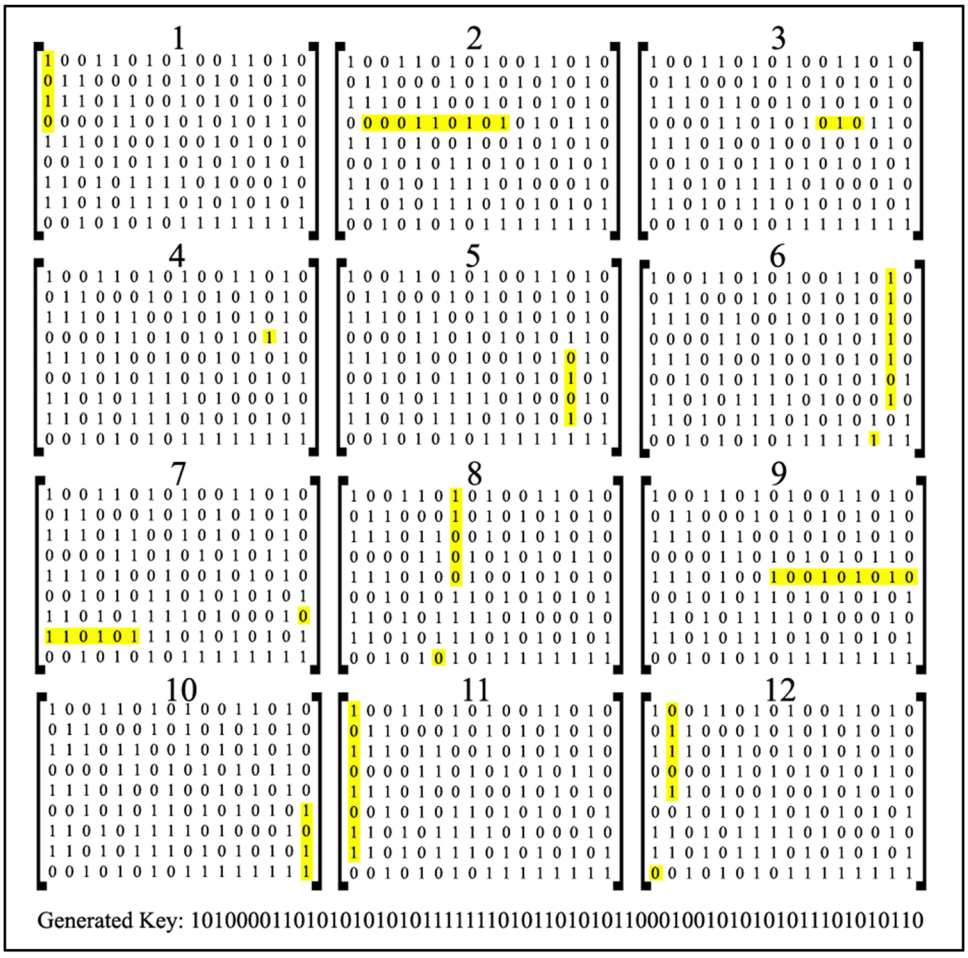

Figure 4: Illustration of the matrix key extraction technique for 493148769486 digits.

the attacker's machine. Applying this feature depends on the end-users and is not applicable in all cases.

2.3 Double valid keys period

There are two valid keys within the first seconds of key refreshment: the current key and the previous key. This double valid keys period is because some packets might be encrypted and sent at the border of changing the key and get delivered when a new key is utilized. Therefore, these packets are encrypted with the previous key, while trying to be decrypted using the current key. To avoid this conflict, in the first seconds of changing the key, the received packets are decrypted with the current key. If the process was unsuccessful, the previous key will be used for decryption. The length of the double valid keys period depends on the maximum required time for sending a packet between the two farthest nodes.

2.4 Joining and leaving

To join the sessions for the first time, the node must pass the authentication process, and then receive the interpreter decryption key and the bit-stream download address. Upon receiving the key, the node is able to decrypt the interpreter and deploy it to establish secure sessions.

If a node has been inactive for a while, it first sends an encrypted test message to either one of the nodes or the server. Receiving a reply means that the interpreter is valid. Otherwise, the interpreter is expired and the node goes through the processes of authentication and downloading a new interpreter again (Figure 5).

When a node decides to leave the sessions permanently, for a reliable node, it is enough to remove the interpreter. However, if any of the disjoined nodes is unreliable, the server revokes all of the distributed interpreters. Therefore, a new version of the interpreter must always be ready in the server for a periodical update or emergency circumstances.

2.5 Application of SIKM in software security

In addition to utilizing SIKM for establishing secure connections, it can be deployed as a component in software to provide authentication and security together. When the group identity is targeted, it helps to bypass the necessity of entering the username and password to gain access. To this end, it is enough to define the level of access to the resources for the interpreter holders.

At the server-side, the users' clearance is recognized according to the employed decryption key. When a user tries to access the resources, the server utilizes the current keys of different interpreter distributions to decrypt the request. The successful decryption key defines the group identity and accordingly grants access.

2.6 Application

The framework is a proper key management choice when the deployed network consists of reliable nodes that hold secure sessions for a long term – at least 34 fresh keys. For example, SIKM is an ideal choice for securing a network of ATMs. Assuming the overall interactions of each customer takes 1 min, minutely key secures its transactions by a unique key without any additional cost.

As another example, a multinational company may use SIKM to secure the communications between its branches. Using one interpreter lets the users communicate securely. As an advanced setup, the branches may deploy SIKM in pairwise mode and secure the data transmission between every two branches independently. This type of deployment requires $n - 1$ different interpreters for n branches.

3 Security and performance

3.1 Security evaluation

This section studies the security of SIKM in three approaches: how it reacts against some cryptographic attacks, removing the key storage component, and the method of utilizing it for one key per transaction. The following paragraphs discuss these in detail.

Smart attack resistance: During the period of double valid keys, if the received packet is not decipherable with either the current or previous keys, SIKM considers the packet as a sign of replay or an exhaustive key search attack and then reports it to the server for the analysis of nodes' security status. The server examines the received packet with the last few previous keys to identify the attack type. If the received packet is not decipherable, it is considered as a sign of exhaustive key search attack, otherwise, a replay attack. Once an attack is identified, depending on the server decision, the node may continue working or shut down temporarily to observing the security measures. The period of double valid keys and the process of attack identification is illustrated in Figure 6.

Key storage: One of the important key management problems is the existence of key storage, which increases the likelihood of compromising the keys. In the common practices, the keys are stored permanently for future sessions, while in SIKM the keys are changing constantly, produced at application time, and are not stored.

Key per session: Ideally, cryptography uses one fresh key for every transaction. However, due to the cost and hardness of implementation, this idea is not simply applicable. Since in SIKM the nodes produce the keys without imposing cost or traffic on the network, a fresh key is utilized in short intervals without imposing extra cost. For instance, by deploying SIKM's minutely key in an ATM network, assuming each customer spends 60 seconds for a transaction, every transaction is secured using a unique key.

3.2 Performance evaluation

To measure the cost-effectiveness of the designed framework, it is compared with the key per session method of the master-key scheme. To this end, equations (1)–(3) help to make a comparison between the key per session

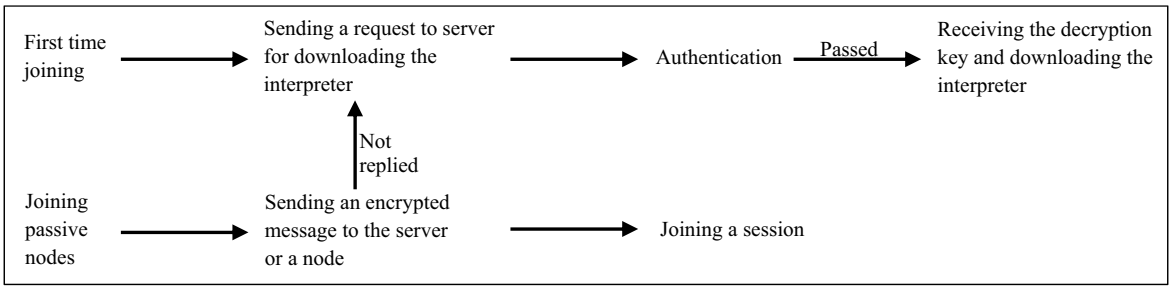

Figure 5: Process of joining sessions.

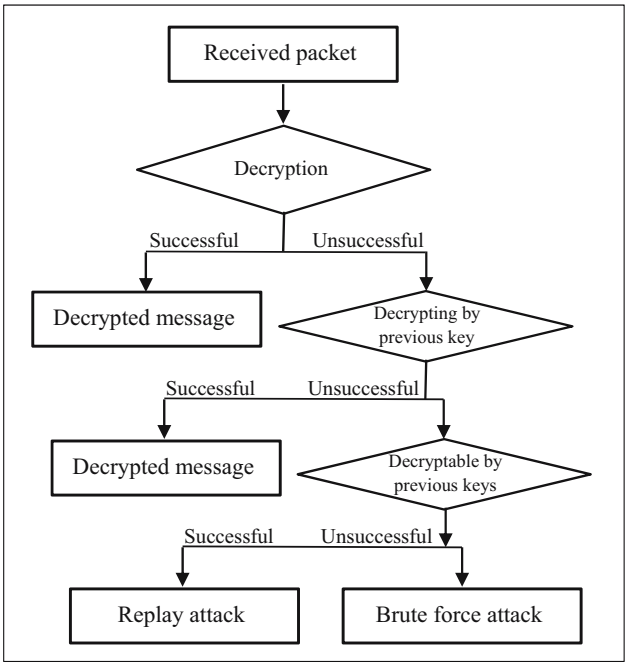

Figure 6: Double key checking and attack identification diagram.

(KPS) scheme and SIKM for minutely, hourly, and daily key refreshment intervals. Table 1 describes the symbols and their corresponding metrics that are used in the equations. The bit-stream refreshment interval is considered once a week.

Equation (1) calculates the imposed key per session traffic for n weeks. Equation (2) computes the traffic of the server activities from the time of establishing SIKM until revoking the interpreter. The total SIKM imposed traffic on the network – which also consists of the server's traffic – is calculated in equation (3).

$$\text{KPS traffic for } n_w = (k_{dl} + k_{rl}) \times n_{kpd} \times 7 \times n_w. \quad (1)$$

$$\text{SIKM traffic for } n_w = (i_{kdl} + i_{irl}). \quad (2)$$

$$\text{SIKM netwotk traffic for } n_w = (i_{kdl} + i_{irl} + i_{idl}) + (n_w \times i_{brl}). \quad (3)$$

Table 1: Traffic of activities

Description	Metric (KB)	Symbol
Number of keys per day	—	n_{kpd}
KPS key distribution traffic	1	k_{dl}
KPS key revocation traffic	0.5	k_{rl}
SIKM key distribution traffic	1	i_{kdl}
SIKM interpreter downloading traffic	50	i_{idl}
SIKM interpreter revocation traffic	0.5	i_{irl}
SIKM bit-stream reloading traffic	0.5	i_{brl}
Number of weeks		n_w

Table 2: The imposed traffic on KPS server and network for producing minutely, hourly, and daily keys

Weeks	No. of keys	Server traf. (KB)	Net traf. (KB)
Minutely key (1,440 keys per day)			
1	10,080	15,120	15,120
4	40,320	60,480	60,480
13	131,040	196,560	196,560
26	262,080	393,120	393,120
52	524,160	786,240	786,240
Hourly key (24 keys per day)			
1	168	252	252
4	672	1,008	1,008
13	2,184	3,276	3,276
26	4,368	6,552	6,552
52	8,736	13,104	1,3104
Daily key (1 key per day)			
1	7	10.5	10.5
4	28	42	42
13	91	136.5	136.5
26	182	273	273
52	364	546	546

Since all of the KPS key distribution traffic is imposed on the server, this traffic is equal for the server and network. However, in SIKM, the traffic is divided into two parts: between the server and the nodes, and between the defined source on the network and the nodes. The traffic of KPS and SIKM for producing minutely, hourly and daily keys for up to 52 weeks is calculated in Tables 2 and 3. To illustrate the differences, Figures 7–9 depict

Table 3: The imposed traffic on SIKM server and network for producing minutely, hourly, and daily keys

Weeks	No. of keys	Server traf.(KB)	Net traf.(KB)
Minutely key (1,440 keys per day)			
1	10,080	1.5	52
4	40,320	1.5	53.5
13	131,040	1.5	58
26	262,080	1.5	64.5
52	524,160	1.5	77.5
Hourly key (24 keys per day)			
1	168	1.5	52
4	672	1.5	53.5
13	2,184	1.5	58
26	4,368	1.5	64.5
52	8,736	1.5	77.5
Daily key (1 key per day)			
1	7	1.5	52
4	28	1.5	53.5
13	91	1.5	58
26	182	1.5	64.5
52	364	1.5	77.5

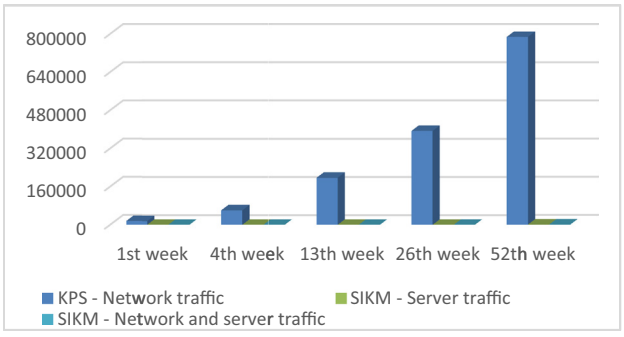

Figure 7: Comparison of KPS and SIKM minutely key traffic on network and server.

the imposed traffic on the KPS server, SIKM server, and SIKM network for minutely, hourly, and daily keys, respectively.

The traffic per key for both KPS and SIKM schemes is calculated in Tables 4 and 5. The tables demonstrate that SIKM daily key imposes just $\frac{1}{7}$ of KPS traffic on the network. For hourly key, the distance increases to $\frac{1}{170}$, and finally, the traffic in SIKM for minutely keys is $\frac{1}{10,714}$ of KPS traffic per key. This means having less than one-bit traffic on the network for every minutely SIKM key.

At the start of establishing SIKM, referring to Table 1 values, 51 KB traffic is generated for receiving the decryption key and then downloading the interpreter. To find the threshold for choosing between KPS and SIKM, the 51 KB of SIKM traffic must be divided into 1.5 KB traffic of distributing and revoking a KPS key. The results show that requiring more than 34 fresh keys is the threshold for employing SIKM instead of KPS. In other words, for daily keys after 34 days, for hourly keys after 34 h, and for minutely keys after 34 min, using SIKM is more cost-effective.

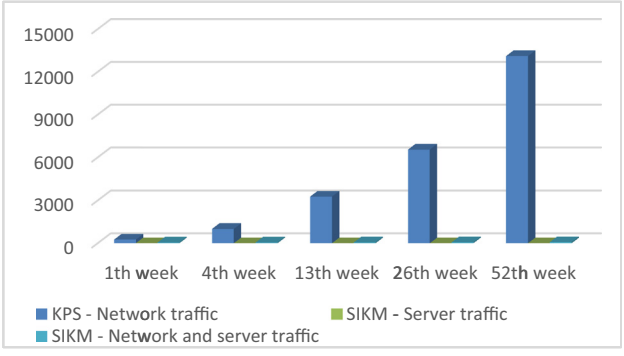

Figure 8: Comparison of KPS and SIKM hourly key traffic on network and server.

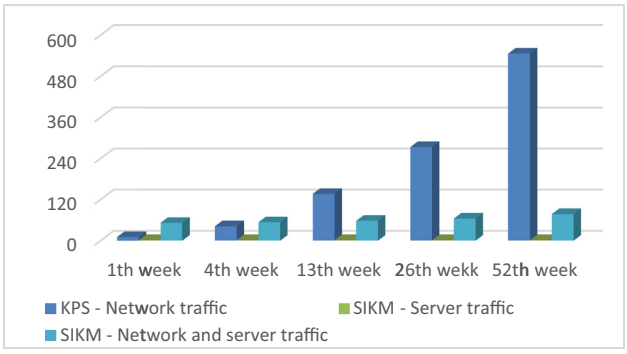

Figure 9: Comparison of KPS and SIKM daily key traffic on network and server.

Table 4: Traffic per key for KPS minutely, hourly, and daily keys after 52 weeks

No. of keys	Net traffic	Net traffic per key
Key per minute		
524,160	786,240 KB	1.5 KB
Key per hour		
No. of keys	Net traffic	Net traffic per key
8,736	13,104 KB	1.5 KB
Key per day		
No. of keys	Net traffic	Net traffic per key
364	546 KB	1.5 KB

Figure 10 depicts the comparison of the generated traffic per key between SIKM and KPS after 52 weeks, and Table 6 compares the main key management features between SIKM and KPS. As shown, KPS has less node workload and the easier process for node revocation, while SIKM outperforms in the rest of features.

Although the proposed framework reduces the cost and traffic load of deploying fresh keys, it has two main drawbacks: first, protecting the distribution of the interpreter decryption key against a man-in-the-middle attack

Table 5: Traffic per key for SIKM minutely, hourly, and daily keys after 52 weeks

No. of keys	Net traffic	Net traffic per key
Key per minute		
524,160	77.5 KB	0.00014 KB
Key per hour		
No. of keys	Net traffic	Net traffic per key
8,736	77.5 KB	0.0088KB
Key per day		
No. of keys	Net traffic	Net traffic per key
364	77.5 KB	0.2129 KB

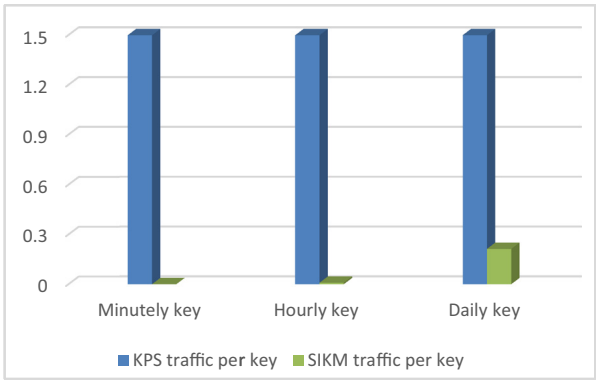

Figure 10: Comparison of network traffic between KPS and SIKM after 52 weeks.

Table 6: Comparing SIKM and KPS features

Features	SIKM	KPS
Low network traffic	✓	X
Distributed computations	✓	X
Low server workload	✓	X
Low node workload	X	✓
Resistant to brute force attack	✓	X
Resistant to replay attack	✓	X
Cost-effectiveness	✓	X
Node revocation	X	✓

and second, the process of revoking a node. If the interpreter decryption key compromises during distribution between any of the nodes, the whole network compromises accordingly, and the server must distribute a new interpreter. However, this is not limited to the SIKM and is a common problem between all of the current practices. Moreover, when a node revokes, the rest must go through the process of receiving and deploying a new interpreter, unless it has been revoked securely.

4 Conclusion

The cyberspace is an online environment where its participants are involved in various types of interactions. The essence of these interactions is the data that is exchanged between the engaged parties. Disclosure of data to unauthorized people is a critical concern for the security of communications, and cryptography is the main solution for overcoming confidentiality threats. Therefore, a range of cryptographic key management frameworks are devised to produce, distribute, and revoke the required keys. However, each one is suitable for specific applications and has its own drawbacks. This article proposes a novel cryptographic key management framework that is cost-effective when more than 34 fresh and dynamic keys are required. It produces the fresh keys in minutely, hourly, or daily intervals and dramatically reduces the traffic load per key in comparison with the master key scheme. In addition, the framework reacts smartly against some cryptographic attacks and eliminates the need for key storage. It is also applicable for authentication in secure sessions when group membership is enough for gaining access.

Conflict of interest: Authors state no conflict of interest.

References

[1] F. Gandino, C. Celozzi, and M. Rebaudengo, "A key management scheme for mobile wireless sensor networks," *Appl. Sci.*, vol. 7, no. 5, p. 490, 2017.

[2] E. Barker, W. Barker, *Recommendation for key management, part 2: best practices for key management organization*, Technical report, National Institute of Standards and Technology, 2018.

[3] E. Yuan and L. Wang, "A key management scheme realising location privacy protection for heterogeneous wireless sensor networks," *Int. J. Sensor Netw.*, vol. 32, no. 1, pp. 34–41, 2020.

[4] G. Xu, X.-B. Chen, Z. Dou, Y.-X. Yang, and Z. Li, "A novel protocol for multiparty quantum key management," *Quantum Inform. Process.*, vol. 14, no. 8, pp. 2959–2980, 2015.

[5] S. S. Chaeikar, H. S. Moghaddam, and H. R. Zeidanloo, "Node based interpretative key management framework," in: *Security and Management*, Las Vegas, USA: WORLDCOMP, 2010, pp. 204–210.

[6] J. Han and J. Wang, "An enhanced key management scheme for LoRaWAN," *Cryptography*, vol. 2, no. 4, pp. 34, 2018.

[7] A. Mazin, K. Davaslioglu, and R. D. Gitlin, "Secure key management for 5g physical layer security," in: *2017 IEEE 18th Wireless and Microwave Technology Conference* (*WAMICON*), Cocoa Beach, FL, USA: IEEE, 2017, pp. 1–5.

[8] J. Liu, X. Tong, Z. Wang, M. Zhang, and J. Ma, "A centralized key management scheme based on mceliece pkc for space network," *IEEE Access*, vol. 8, pp. 42708–42719, 2020.

[9] S. S. Chaeikar, S. Yazdanpanah, and N. S. Chaeikar, "Secure sms transmission based on social network messages," *Int. J. Internet Technol. Secured Trans.*, vol. 11, no. 2, pp. 176–192, 2021.

[10] S. R. Singh and K. K. Ajoy, "Key management scheme for internet of things using an elliptic curve," *J. Comput. Theoret. Nanosci.*, vol. 17, no. 1, pp. 115–121, 2020.

[11] G. Manikandan and U. Sakthi, "A comprehensive survey on various key management schemes in WSN," in: *2018 2nd International Conference on I-SMAC* (*IoT in Social, Mobile, Analytics and Cloud*)(*I-SMAC*) *I-SMAC* (*IoT in Social, Mobile, Analytics and Cloud*)(*I-SMAC*), Palladam, India: IEEE, 2018, pp. 378–383.

[12] A. Joshi and A. K. Mohapatra, "Authentication protocols for wireless body area network with key management approach,"

J. Discrete Math. Sci. Cryptograph., vol. 22, no. 2, pp. 219–240, 2019.
[13] Y. Tian, Z. Wang, J. Xiong, and J. Ma, "A blockchain-based secure key management scheme with trustworthiness in DWSNs," *IEEE Trans. Industr. Inform.*, vol. 16, no. 9, pp. 6193–6202, 2020.
[14] M. Alizadeh, M. Salleh, M. Zamani, J. Shayan, S. Karamizadeh, *Security and performance evaluation of lightweight cryptographic algorithms in RFID*, The 16th WSEAS International Conference on Communications (part of CSCC '12), Kos Island, Greece, July 14–17, 2012.
[15] J. Shen, H. Tan, S. Moh, I. Chung, Q. Liu, and X. Sun, "Enhanced secure sensor association and key management in wireless body area networks," *J. Commun. Netw.*, vol. 17, no. 5, pp. 453–462, 2015.
[16] L. Zhang, "Key management scheme for secure channel establishment in fog computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1117–28, 2019.
[17] S. ShojaeChaeikar, A. A. Manaf, A. A. Alarood, and M. Zamani, "PFW: polygonal fuzzy weighted an svm kernel for the classification of overlapping data groups," *Electronics*, vol. 9, no. 4, p. 615, 2020.
[18] Y. Harchol, I. Abraham, and B. Pinkas, "Distributed ssh key management with proactive rsa threshold signatures," in: *International Conference on Applied Cryptography and Network Security*, Leuven, Belgium: Springer, 2018, pp. 22–43.
[19] A. Ghosal and M. Conti, "Key management systems for smart grid advanced metering infrastructure: A survey," *IEEE Commun. Surveys Tutorials*, vol. 21, no. 3, pp. 2831–2848, 2019.
[20] S. S. Chaeikar, M. Alizadeh, M. H. Tadayon, and A. Jolfaei, "An intelligent cryptographic key management model for secure communications in distributed industrial intelligent systems," *Int. J. Intell. Syst.*, 2021.
[21] M. Ma, G. Shi, and F. Li, "Privacy-oriented blockchain-based distributed key management architecture for hierarchical access control in the iot scenario," *IEEE Access*, vol. 7, pp. 34045–34059, 2019.
[22] J. Sen, "Cryptography and security in computing," *BoD–Books on Demand*, London: IntechOpen, 2012.
[23] S. S. Chaeikar, A. Jolfaei, N. Mohammad, and P. Ostovari, "Security principles and challenges in electronic voting," in: *2021 IEEE 25th International Enterprise Distributed Object Computing Workshop* (*EDOCW*), Gold Coast, Australia: IEEE, 2021, pp. 38–45.
[24] M. S. Yousefpoor and H. Barati, "Dynamic key management algorithms in wireless sensor networks: A survey," *Comput. Commun.*, vol. 134, pp. 52–69, 2019.
[25] T. Pramod, K. G. Boroojeni, M. H. Amini, N. Sunitha, and S. Iyengar, "Key pre-distribution scheme with join leave support for scada systems," *Int J. Critic. Infrastruct. Protect.*, vol. 24, pp. 111–125, 2019.
[26] S. Yazdanpanah and S. S. Chaeikar, "IKM-based security usability enhancement model," *Int. J. Comput. Sci. Inf. Technol. Secur.*, vol. 2, no. 4, pp. 852–858, 2012.
[27] L. Li, G. Xu, L. Jiao, X. Li, H. Wang, J. Hu, et al., "A secure random key distribution scheme against node replication attacks in industrial wireless sensor systems," *IEEE Trans. Ind. Inform.*, vol. 16, no. 3, pp. 2091–2101, 2019.
[28] S. Mandal, S. Mohanty, and B. Majhi, "CL-AGKA: Certificateless authenticated group key agreement protocol for mobile networks," *Wireless Netw.*, vol. 26, pp. 3011–3031, 2020. Doi: 10.1007/s11276-020-02252-z.
[29] S. S. Chaeikar, A. B. A. Manaf, and M. Zamani, "Comparative analysis of master-key and interpretative key management (IKM) frameworks," *Cryptograph. Security Comput.*, vol. 203, pp. 203–218, 2012.
[30] R. L. Naik, S. S. S. Reddy, and M. G. Chand, "Toward secure quantum key distribution protocol for super dense coding attack: A hybrid approach," in: *Data Engineering and Communication Technology*, Singapore: Springer, 2020, pp. 515–525.
[31] M. Griotti, F. Gandino, and M. Rebaudengo, "Transitory master key transport layer security for WSNS," *IEEE Access*, vol. 8, pp. 20304–20312, 2020.
[32] S. S. Chaeikar, S. AbdRazak, S. Honarbakhsh, H. R. Zeidanloo, M. Zamani, and F. Jaryani, "Interpretative key management (IKM), a novel framework," in: *2010 Second International Conference on Computer Research and Development*, Kuala Lumpur, Malaysia: IEEE, 2010, pp. 265–269.