Journal of Engg. Research Vol. 8 No. (3) September 2020 pp. 91-117 https://doi.org/10.36909/jer.v8i3.8079

تحسين نظام مشاركة السرية القائم على العد بتوليد المشاركات السرية بعد التقسيم الكتلي بطريقة أكثر أماناً

عدنان بن عبدالعزيز قطب وعادل بن محمد القرشي
قسم هندسة الحاسب الآلي، جامعة أم القرى، مكة المكرمة، المملكة العربية السعودية

الخلاصة

تم تقديم نظام مشاركة السرية القائم على العد مؤخراً كنهج واعد يخدم التطبيقات التي تتطلب مصادقة عدد من المستخدمين. حيث يعمل المخطط في الأصل على توليد المشاركات من خلال عملية تقليب بسيط لواحد أو اثنين من البتات داخل المفتاح السري بأكمله في مواقع مختلفة. وعند الحاجة إلى استرجاع المفتاح السري يتم جمع المشاركات المختارة على أساس العتبة (ن، ك)، ومن ثم تُطبق طريقة العد الخاص بالتوازي على هذه المشاركات لاستعادة المفتاح السري. يقترح هذا البحث تعديل عملية توليد المشاركات من أجل تعزيز الأمان، عن طريق تقسيم المفتاح السري إلى كتل حيث تتضمن كل كتلة تقليب بتات محددة في وقت واحد، مما يؤدي إلى توليد مشاركات غامضة تعمل على تحسين أمان نظام الوصول. تم تنفيذ طرق توليد المشاركات بناءً على الكتل في نماذج مختلفة من 64 بت عبر منصة جافا للحصول على اختبار عادل. حيث أظهرت التجارب نتائج ومقارنات مثيرة للاهتمام تقدم اسهامات أمنية رائعة، ويمكن اعتبار البحث اتجاهً مفتوحاً لإجراء المزيد من الأبحاث الجذابة التي تسهم في تعزيز نظام المشاركة السرية القائم على العد.

Secure Shares Generation via M-Blocks Partitioning for Counting-Based Secret Sharing

Adnan Gutub*,** and Adel Al-Qurashi*

*Computer Engineering Department, Umm Al-Qura University, Makkah, Saudi Arabia
**Corresponding Author: aagutub@uqu.edu.sa

Submitted: 23/03/2019
Revised: 04/11/2019
Accepted: 12/11/2019

ABSTRACT

Counting-based secret sharing is presented recently as a promising approach serving multiuser authentication applications. The scheme originally generates its shares via simple flipping of one or two 0-bits within the entire secret key at various locations. Reconstructing the secret key combines chosen shares, based on (n,k) threshold, in parallel specific counting to recover back the secret key. This paper proposes modifying the shares generation process, for security enhancement, by dividing the secret key into blocks. Each block involves flipping specific bits simultaneously, generating ambiguous shares improving the access system security. The proposed blocks flipping shares methods are implemented in different 64-bits models via fair testing Java platform. Experimentations showed interesting comparisons results providing remarkable secure contributions. The work can be considered an opening applicability direction for further attractive research in improving the counting-based secret sharing technique.

Keywords: Counting-based secret sharing; Shares generation; Shares construction; Key management; Key distribution; Information security.

INTRODUCTION

Nowadays, the demand for information security has increased. Its objective did not change, to maintain confidentiality, integrity, and availability, but with more new usages. The security of data within organizations against threats is becoming more important than before (Gutub et al., 2017). Consequently, information security techniques have appeared protecting data from unethical disclosure via cryptography or steganography (Alassaf et al., 2018). However, in these techniques (cryptography and steganography (Alsaidi et al., 2018)) there is normally a single person controlling access and in charge of security of the data within the system (Alaseri et al., 2018). The problem arises if the encryption key for the encrypted or hidden data is lost. This indicates that there is only security, but without reliability (AlQurashi et al., 2018). In fact, losing or misusing the encryption key leads to loss of data or its benefit, making access to secured data almost impossible (Al-Juaid et al., 2018). Therefore, keeping the encryption key with one person is not dependable especially when the system is exposed to any physical or electronic problem. So, this key management centralized access process is addressed by distributing copies of the encryption key to more than one person allowing sharing the control power. The intention leads to increasing the reliability and multiple access to the system, but may reduce confidentiality, perhaps making the security status worst where data may be exposed to great risk to be lost, modified, destroyed, or leaked to wrong hands (Binu et al., 2016). Accordingly, the secret sharing scheme came into picture considering solving these problems. A lot of researches have proposed ideas about sharing a secret among a set of participants, where successful secret sharing became possible, i.e., allowing to achieve high levels of confidentiality and reliability. This research of secret sharing is making the secret key collectively controlled by a set of participants (Shamir, 2006), avoiding the

dominance of a specific authority and the individual trusty. The main philosophy of this secret sharing depends on keeping the secret key from hands of solo one person to control the secrecy of information (Binu et al., 2016). In addition, the secret sharing allows collective prioritization access as well as partial user's access to the secret information, leading to the joint decision benefiting from distributing trust among many participants. In fact, secret sharing is believed to enhance the confidentiality and reliability of the process to access sensitive applications and resources (AlQurashi et al., 2018). It forces the requirement access to be multiparty agreed upon needed by the sensitive information and resources linked to big impact on decision making, such as voting systems, nuclear missile launch control, opening the vault in central banks, medical agreement, and sensitive encryption keys (Gutub et al., 2017).

The secret sharing scheme divides a secret among a group of participants, where a specific group can recover the secret. Therefore, this secret sharing technique involves two stages: shares construction and distribution stage, and secret reconstruction stage. In the stage of shares construction and distribution, the system controller or dealer, assumed to be fully trusted, constructs the shares from the secret key by making some of its data-bits changed, i.e., as shares. Then, the shares are to be distributed among the intended group of the participants, as in Fig. 1, via secure channel or key management system (Al-Juaid et al., 2018). The reconstruction stage allows the qualified subgroup of contributors to cooperate and restructure the secret key by gathering the shares in particular (Iftene, 2006).

Considering Fig.1, the secret sharing organization distributes the shares among (n) participants. However, the system does not allow all participants to need to restructure the secret key (Kaya, 2009) by keeping approved set (k) as a subgroup of (n) contributors (k out of n) to be sufficient. Note that (k) indicates to be less than or equal to (n) participants ($k \leq n$). Thus, the secret sharing system will be gathering their shares by specific way to retrieve the secret key, which is based on a threshold of the secret sharing scheme (Iftene, 2006).

In the threshold secret sharing system, the secret key must be very secure, where attackers cannot guess the secret key, even when knowing the shares (AlQurashi et al., 2018). In general, the secret sharing schemes normally require two main properties to be accepted, namely, confidentiality and recoverability, as defined below.

- Confidentiality: is that shares do not contain any information about the secret hidden key, i.e., not allowing any hacker effort to make progress finding secret key by predicting from less than k shares.
- Recoverability: the ability of the authorized set of contributors to recover the secret key by gathering intended shares.

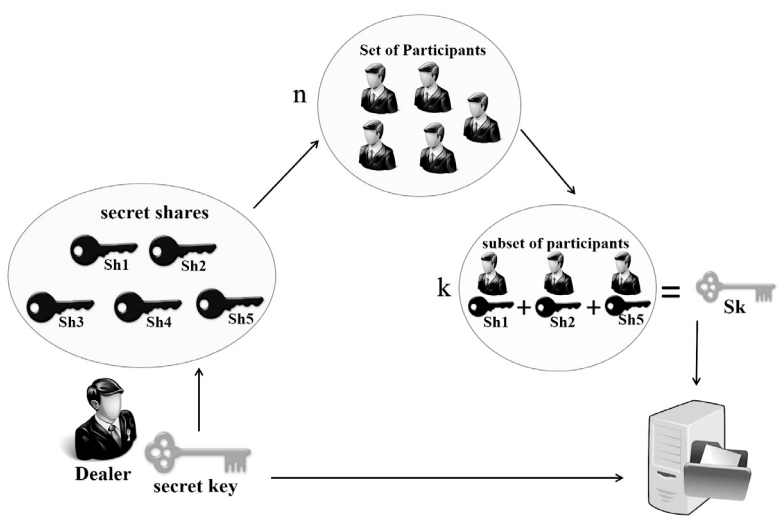

Figure 1. The notion of Threshold Secret Sharing Outline.

In this research, we will present refining the security of the Counting Based Secret Sharing Scheme (CBSSS) innovated recently by Adnan Gutub (2017). The motivation came from studying CBSSS extensive variety of applicability to almost all secret sharing uses, as well as its straightforwardness in its implementation and utilization. This CBSSS works first on generating shares by two original methods, namely, 1-bit & 2-bits methods. Then, the grouping of the shares can be applied via matching counting within selected shares (k shares) to recover the secret key.

This paper proposed improving the security of counting-based secret sharing scheme CBSSS by raising the size of the secret key to 64-bits as recommended minimum required secure key-size for most applications (Al-Ghamdi et al., 2018). In addition, we proposed a new method for generating shares in CBSSS based on blocks method. The secret key is divided into blocks of M-bits in size, where every block is altered through the original sharing generation techniques, i.e., 1-bit & 2-bit methods, to generate the new shares. This block secret key modification to generate shares is showing variation in improving the security of shares, but limiting the total number of shares generated per secret key as will be detailed in the study.

The paper has been organized as follows. Section 2 covers the related work about secret sharing systems. Section 3 presents specific background needed from the original counting-based secret sharing scheme. Section 4 presents our proposed modeling enhancement for shares security improvement within CBSSS. Section 5 discusses the comparisons of security study for the different models changing the block sizes analyzing the results. Section 6 presents comparisons with other related works connected to CBSSS. The final section, Section 7, includes the conclusion and some recommendations for future research continuing this study.

RELATED WORK

The threshold secret sharing scheme is originally proposed by Shamir (2006) and Blakley (1979), independently in the same year. Since then, a lot of research work has been proposed about different threshold secret sharing schemes, where many research works in the literature discussed threshold secret sharing from several aspects. Some focused on the techniques used, the number of shared secrets, share weight, the changeability of shares, and the rights given to users.

Lately, Bai et al. (2009) enhanced Shamir's single-secret sharing pattern to multiple-secret sharing structure using matrix projection. They suggested an active secret sharing model that renews (n) secret shares occasionally in a (k, n) threshold-based secret sharing scheme, deprived of changing the secret, or rebuilding the secret to creating new shares. The work of Bai et al. (2009) offered a scattered practical secret sharing model for the matrix projection secret sharing technique. Note that their organization cannot expose the secrets from (k) shares by challengers when new shares are updated, which has been mixed with previous and current shares, i.e., making this method threatened against the unreceptive attacks.

Beimel et al. (2005) considered weighted threshold secret sharing. They presented that weighted threshold contact assembly is possible if and only if it is a ranked threshold entree organization, or a triple access structure, or a structure of two ideal weighted threshold access structures, which are determined on smaller sets of users. Through all those cases, the weighted threshold access structure may be achieved by a linear ideal secret sharing scheme. Morillo et al. (1999) dealt with weighted threshold schemes, through the property of information rate. The access structures of weighted threshold schemes presented complete characterization of all the minimal authorized subsets that have involved at most two elements. Lower bounds for the optimal information rate of these access structures are given and can be further referred to as clarified in Morillo et al. (1999).

In multilevel organizations, there is the demand to share a secret among all the users of the organization in hierarchically structured groups (Castiglione et al., 2014). Sharing of data is constructed on a prearranged categorization of threshold requests. Such thresholds necessitate the existence of a participant with upper level to develop the organization's secret, as in Tamir-Tassa's Hierarchical Secret Sharing (Tassa, 2007). Tamir-Tassa's discussed the secret sharing model to adopted polynomial derivatives to produce few shares for contributors of subordinate levels, as the unrestricted coefficient of some polynomial.

In Mignotte's threshold secret sharing scheme (Mignotte, 1982), the research relied on modulo arithmetic and Chinese Remainder Theorem (CRT) making it more complex for applicability as well as giving small shares. The scheme has been improved by familiarizing Mingotte sequences as comprehensive, assuming elements are unessentially pairwise coprime values as elaborated in (Sorin et al., 2007).

Taghrid Al-Khodaidi et al. (2019) presented a scalable shares generation scheme to increase participants using the counting-based secret sharing technique. The work of Al-Khodaidi et al. (2019) suggested seven models (Add at first, Add at middle, Add at the end, 1R1Z, 2R2Z, 1R1Z1R2Z, and 1R2Z2R1Z) to produce different versions of secret keys SK that allows the system to choose after comparison. All seven secret keys SK models are compared in parallel while the system is running to select the best based on measuring security degree, i.e., to find the best security of the secret key to use in generating all possible shares.

Similarly, Maimoona Al-Ghamdi et al. (2018) presented a security enhancement method serving counting-based secret sharing. She improved the shares generation process for multimedia usages affecting the original Gutub' scheme (Gutub et al., 2017) by controlling different disadvantages in the effectiveness relating the measurement and construction of the secret key as well as the number of generated shares. The research suggested four models for the secret key generation with different sizes. They further proposed adjusting any key size to ensure reaching the appropriate unified length adopted in the secret key security study. Their research can be summarized presenting two adjusted models for shares generation further to the basic generation method. The work focused on considering the avoidable cases within Gutub (2017) work observing the number of zero-bits less than one-bit, i.e., within the secret key, making the algorithm generate the shares via novel shares generation methods opposite to the known original work. Both related counting-based secret sharing works of Taghrid Al-Khodaidi (2019) as well as Maimoona Al-Ghamdi (2018) will be considered for comparisons to others, as presented later in Section 6.

BACKGROUND OF COUNTING-BASED SECRET SHARING

The counting-based secret sharing scheme (CBSSS) mainly builds its shares generation by using two methods, namely, 1-bit & 2-bits methods, which both work in different styles to generate all the conceivable shares of the secret key SK denoted by 'A' shares (Gutub et al., 2017). These 'A' shares contain similar length of bits making up the secret key.

In fact, the shares are basically SK but with change in one or two 0-bits within the entire sequence of SK. The scheme chooses (n) shares accurately from 'A' shares (i.e., n out of A shares), which must be useful and able, when combined, to recover SK, assuming the remaining shares are ignored. These (n) shares are distributed among participants by authentic channel or trusted dealer (Fig. 1). Nonetheless, in case one or more of (n) shares are absent or unavailable, SK cannot be retrieved. Therefore, it requires providing a subset of (n) shares denoted by (k) shares where ($k \leq n$) is able to recover SK, as observed in the diagram in Fig. 2. The CBSSS can be categorized as (n, k) threshold scheme. It is to be mentioned that the security of any system depends on the difficulty of recovering SK from shares less than (k) (Gutub et al., 2017).

This section will present the original two methods 1-bit & 2-bit to generate shares, which produce the pool of shares 'A' consisting of acceptable and unacceptable shares and then present the mechanism to recover the secret key SK from the shares, which will be briefed while clarifying the counting-based secret sharing approach.

Shares Generation via 1-Bit Method

All original shares generated via the 1-bit method depend mainly on one 0-bits within SK entire sequence. This method selects one 0-bits from every specific location of secret key SK for flipping to 1-bit to produce a valid share. Every flipping within SK different location is generating a new share; i.e., it must be selecting various positions in chain SK not previously selected for producing another share and so forth (Gutub et al., 2017).

The example in Table 1 clarifies the 1-bit method to generate shares, where the size of SK is 8-bits and SK= [10100001], which is represented in Hexadecimal as SK= (85) hex. Note that, in this example, SK has five zero-bits.

Therefore, it can generate only five shares by changing one 0-bits to 1-bit, as in the highlighted bits in Table 1. It is noted that, in this 1-bit method, all shares are useful for performing the parallel counting to recover the secret key, but it gives the limited number of shares making us consider the 2-bits method described next.

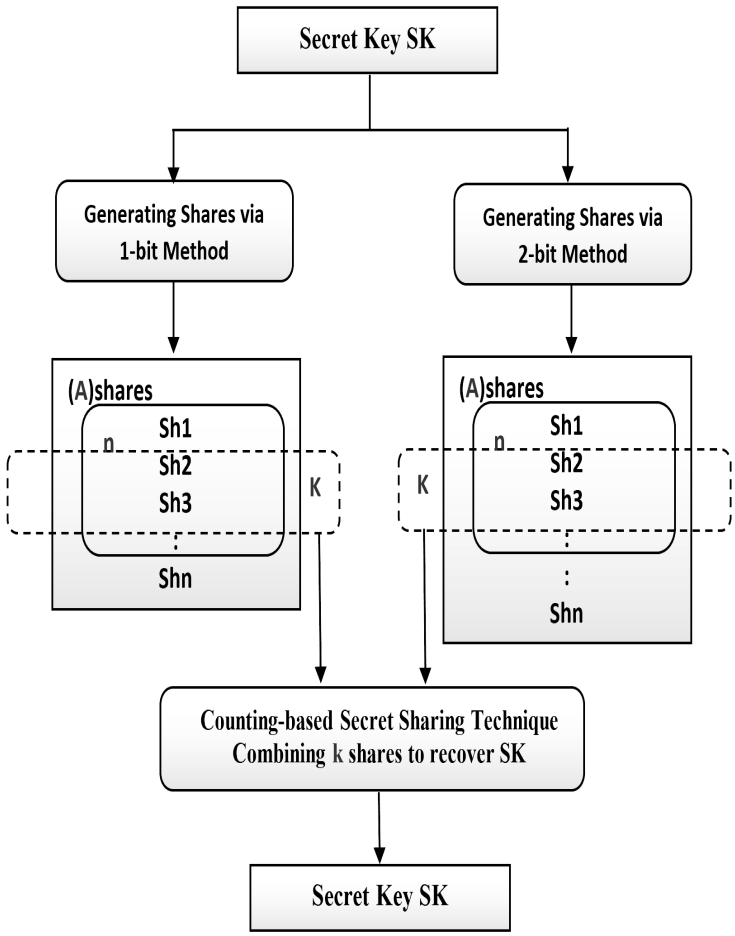

Figure 2. The notion of Counting-Based Secret Sharing Scheme CBSSS [1].

Table 1. Example of the 1-bit method shares generation.

SK	1	0	1	0	0	0	0	1	85 Hex
Sh_1	1	1	1	0	0	0	0	1	87
Sh_2	1	0	1	1	0	0	0	1	8D
Sh_3	1	0	1	0	1	0	0	1	D5
Sh_4	1	0	1	0	0	1	0	1	A5
Sh_5	1	0	1	0	0	0	1	1	C5

Shares Generation via 2-Bits Method

The 2-bits method enhances the 1-bit method to increase the number of shares in CBSSS, which can be used alone or as an extension with the 1-bit method. This method is based on two 0-bits within SK entire sequence to generate extra possible shares. It works by scanning SK sequence looking for zeros; when it finds two 0-bits not previously used together, they are flipped to 1-bits to produce a new share. In fact, not all generated shares by this method are useful, i.e., when applying the counting in parallel to recover the secret key SK. Thus, these shares need to be tested for applicability before distributing to participants. The number of expected shares possibly generated by this 2-bits method can be estimated by the formula: A_{sh}, where (A_{sh}) is the pool 'A' generated shares, while ( ) is the number of zeros within SK sequence.

The following example (extended from Table-1) elaborates the 1-bit method, as shown in Table 2. The 2-bits method generates more shares than before, as in the highlighted bits in Table 2. Note that, in this example, the number of expected generated shares from SK=[10100001] is 10 shares.

Table 2. Example of the 2-bits method shares construction.

SK	1	0	1	0	0	0	0	1	85 Hex
Sh_1	1	1	1	1	0	0	0	1	8F
Sh_2	1	1	1	0	1	0	0	1	97
Sh_3	1	1	1	0	0	1	0	1	A7
Sh_4	1	1	1	0	0	0	1	1	C7
Sh_5	1	0	1	1	1	0	0	1	9D
Sh_6	1	0	1	1	0	1	0	1	AD
Sh_7	1	0	1	1	0	0	1	1	CD
Sh_8	1	0	1	0	1	1	0	1	B5
Sh_9	1	0	1	0	1	0	1	1	D5
Sh_{10}	1	0	1	0	0	1	1	1	E5

Secret Key Reconstruction

The interesting feature of CBSSS is that it does not need to use all 'A' shares to recover SK; as mentioned before, only selected (n) shares are validated to be used as the authorized set from participants. When there is a need to recover the secret key SK, the application determines a value (k) (k out of n) to be used as the threshold of available true users shares for SK reconstruction (Gutub et al., 2017). Therefore, the (k) shares are gathered in parallel within the system, and the parallel bits are calculated. If the counting output from all bits in one column equals the value of (k) or more, then the resulting bit is one; otherwise, the resulting bit is zero, and so on. These resulting bits are combined to reconstruct the secret key, i.e., compared with original SK to check the validity of the shares as approved SK secret key.

The following example, in Fig. 3, illustrates the secret key reconstruction mechanism from shares in CBSSS, SK = [10100001]. The shares have been generated by 1-bit & 2-bits methods from SK, existing in Table 1, and Table 2. Assume the number of selected shares to be given to users is n=10 and threshold K=5. The SK is reconstructed using shares Sh_1, Sh_2, Sh_6, Sh_7, and Sh_{10}, providing counting result of 53511215 giving the correct SK, as shown in Fig. 3.

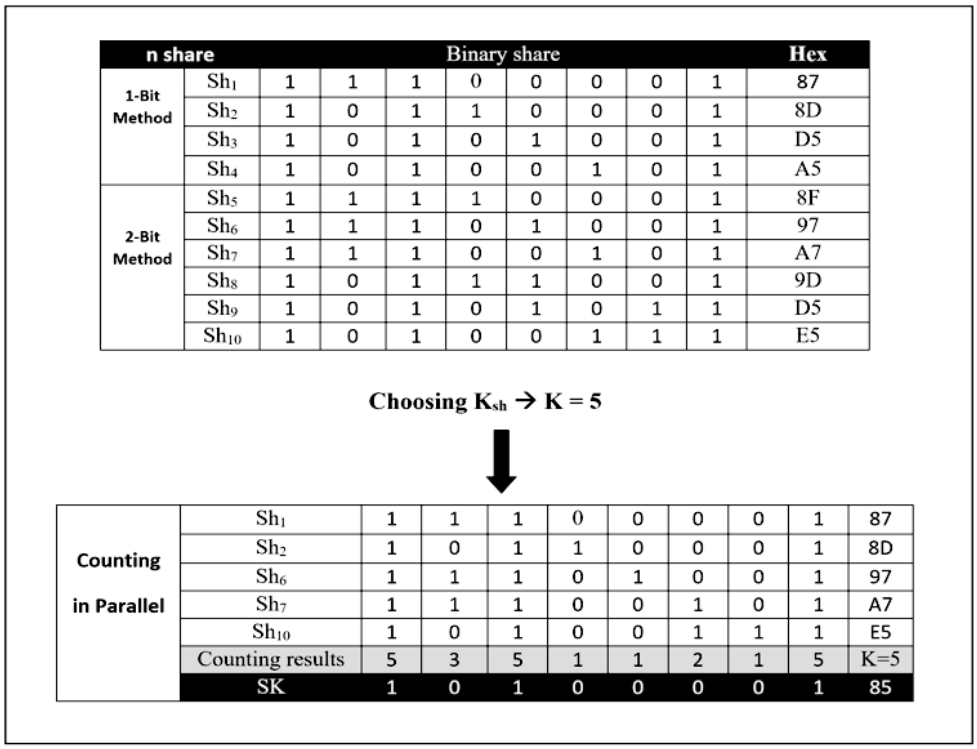

Figure 3. Example of the secret key reconstruction mechanism.

PROPOSED SECURE SHARES GENERATION FOR CBSSS

This research proposes to advance the security of the original counting-based secret sharing scheme CBSSS presented in Gutub et al. (2017) via improving its shares generation techniques. The new CBSSS works in the same way as the previous scheme but with some adjustments. We suggested a new approach of block segmentation when applying the 1-bit and 2-bits methods used to generate the shares from the secret key SK. This scheme can be clarified applying the same two stages, generating shares from the secret key and retrieving secret key from specific shares. In the first stage, the new CBSSS works on generating all the possible shares from secret key SK via the use of improved approach to implement the 1-bit and 2-bits methods within M-blocks. These shares are to be generated from SK with the same size of blocks to ensure appropriate security and reduce the probability of easily guessing SK from shares. Interestingly, the second stage of SK retrieval used the same counting process as the same old CBSSS technique in (Gutub et al., 2017), i.e., after providing the sufficient number of specific shares. The modified proposed CBSSS added the block division process before applying the semioriginal shares generation scheme as illustrated in the algorithm flow graph of Fig. 4. Note that the source of the SK is assumed to be generated randomly through the random number generator RNG pretended available, trusted within the system.

In this CBSSS, Fig. 4, the system dealer (Algorithm) generates SK secret key with size 64-bit by using Random Number Generator (RNG) in binary format, for the aim of producing a trusted secret key SK. The secret key SK should be unknown to all participants. This justifies that SK sequence must be random allowing the difficulty of predicting it by participants or intruders. Therefore, generating the secret key by RNG needs to be verified to be realistic (trusted) before generating shares. In the proposed work, we consider the reliable secret key generation for counting-based secret sharing as clarified in (AlQurashi et al., 2018), which applied two statistical test standards to be checked. The reliability is derived from frequency (Monobit) test and the frequency block test as standard experimentations from NIST 800-22 suites (Rukhin et al., 2001), i.e., to test the randomness of SK getting the reliable random secret key applicable for CBSSS (AlQurashi et al. 2018).

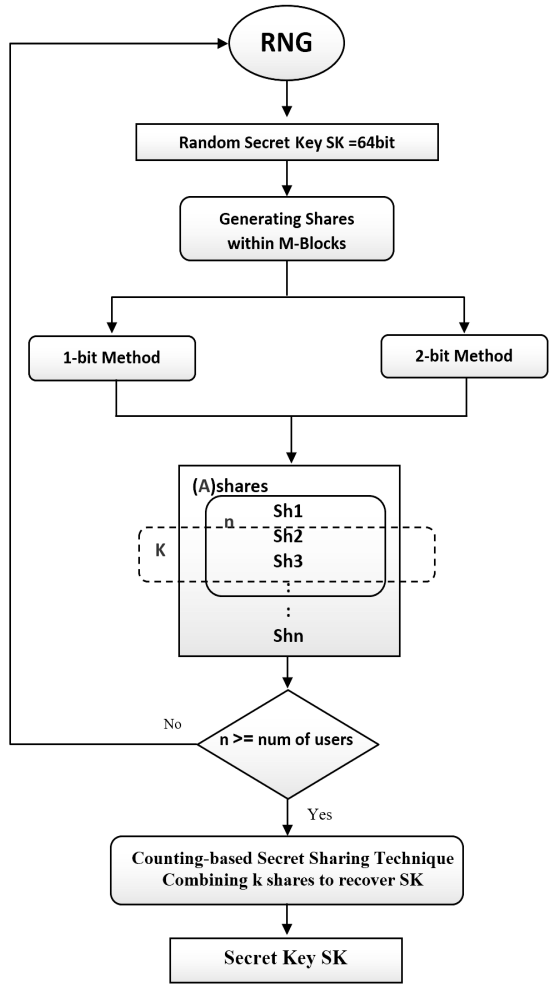

Figure 4. Proposed modified counting-based secret sharing scheme.

In this proposal of new block shares generation method, we simulated the improved algorithm by means of Intel Core i7 processor PC running via 2.90 GHz frequency, on memory of RAM 16 GB, with 64-bit operating system. The platform used is NetBeans IDE version 8.9 as research programming environment for simulating purposes. The research database used is MySQL Workbench version 6.3 as available memory to store results and make them connected to NetBeans IDE. Mining the outputs has been achieved by reading the database MySQL Workbench followed by transferring its lists to Excel program for detailed investigation and assessment. It is to be mentioned that CBSSS Java platform and software programs are geared for testing experimentations and to provide fair comparison study between the models. The software platform is not optimized to the adoption of real-life utilization or commercial usages.

Shares Generation via Secret Key as M-Blocks

In the first CBSSS model, every share is considered to be developed from the secret key SK but with changing one or two 0-bits as previously presented in 1-bit or 2-bits methods (Gutub et al., 2017). This straightforward CBSSS shares generation process relies on 0-bits within SK to generate shares. We increased the size of the secret key proposed in our new scheme as 64-bits, to pretend that the minimum recommended key-size in real-life applications is password numbers (AlQurashi et al., 2018). Accordingly, we proposed this new approach to improve the old method

in generating shares through dividing the size of the secret key to M equal blocks based on security options required within the applications and usages, i.e., assuming M is the number of blocks. Then we apply the 1-bit and 2-bits methods within each block simultaneously to generate the shares, as shown in Fig. 5. The goal of applying this new approach is to increase the shares ambiguity within the system to be securely unable to relate between the shares and secret key. In fact, it is noted clearly that the similarity existence among shares can be also used by intruders making the probability to guess SK from shares originally possible. Therefore, we suggested these new options of M-blocks applying the 1-bit and 2-bits methods to generate the list 'A' shares from SK after dividing SK sequence into equal M-blocks, namely, 2-blocks, 4-blocks, and 8-blocks, as testing models in this study. The simulation will start with 1-block testing, which is the basic original 1-bit method used in the previous scheme (Gutub et al., 2017). Then, we will test dividing the secret in several equal blocks, as shown in Fig. 5. Through these methods, we assume three options for a permitted number of the block to implement the 1-bit and 2-bits methods, which give disparate levels of security and varying capacities of A shares to be available in the applications of CBSSS. The study is based on the number of shares expected to be generated from SK in the 1-bit method within M-block depending on the number of zeros (nz) within SK divided on the number of blocks M (A_{sh} = nz /M), while the number of shares expected to be generated from SK in the 2-bit method within M-block depends on the previously noted summation formula: A_{sh}, where a= nz /M.

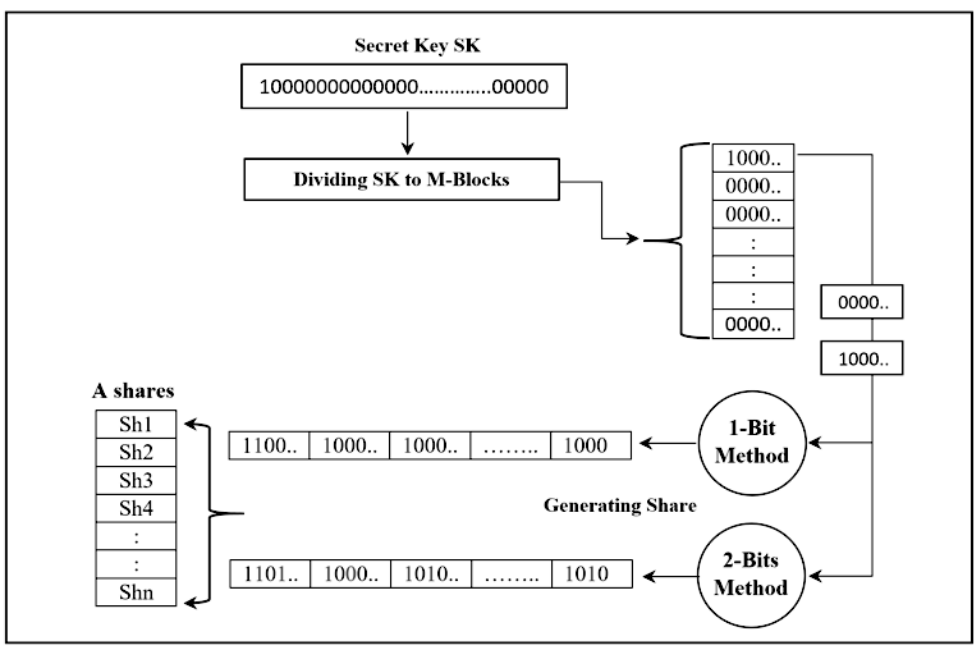

Figure 5. Proposed new method for shares generating based on M-blocks.

Shares Generated from SK as 1-Block

This method is similar in principle to the basic 1-bit and 2-bits method of the CBSSS (Gutub et al., 2017), but with the secret key size improved to 64-bit, as to be used for comparison completion to the newly proposed work. The 1-bit technique will select one 0-bit from a precise location of SK arrangement and then reverse it to 1-bit to yield a valid share. The 2-bits method works by flipping two 0-bits within SK sequence to produce valid shares. Note that, every time in generating the new shares, we must be selecting the various position in SK 64-bit not previously selected, i.e., for producing other shares and so forth. Recall that not all generated shares by the 2-bit method are useful for applying the counting-based parallel secret recovering strategy, unlike the fully approved 1-bit method, to reconstruct the secret key (Gutub et al., 2017). Thus, these 2-bit method shares need to be tested before distributed to participants. The number of expected shares is estimated by the summation formula: A_{sh} + nz.

The following example, as in Table 3, clarifies the 1-bit and 2-bits methods within the 1-block entire SK sequence, i.e., to generate all possible shares. Consider SK as follows.

SK= [10110010 00110101 01101101 00110001 00110110 00110011 01100110 01110010]

Table 3. Example listing samples of shares generated assuming SK as 1-block.

Binary			Hex
	SK	10110010 00110101 01101101 00110001 00110110 00110011 01100110 01110010	**B2 35 6D 31 36 33 66 72**
Method	**A**	Block 1	**B2 35 6D 31 36 33 66 72**
1-Bit	**Sh_1**	1011**1**010 00110101 01101101 00110001 00110110 00110011 01100110 01110010	**BA 35 6D 31 36 33 66 72**
	Sh_2	10110010 **1**0110101 01101101 00110001 00110110 00110011 01100110 01110010	**B2 B5 6D 31 36 33 66 72**
	Sh_3	10110010 00110101 **1**1101101 00110001 00110110 00110011 01100110 01110010	**B2 35 ED 31 36 33 66 72**
	Sh_4	10110010 00110101 011011**1**1 00110001 00110110 00110011 01100110 01110010	**B2 35 6F 31 36 33 66 72**
	Sh_5	10110010 00110101 01101101 0011**1**001 00110110 00110011 01100110 01110010	**B2 35 6D 39 36 33 66 72**
	Sh_6	10110010 00110101 01101101 00110001 0011011**1** 00110011 01100110 01110010	**B2 35 6D 31 37 33 66 72**
	Sh_7	10110010 00110101 01101101 00110001 00110110 **1**0110011 01100110 01110010	**b2 35 6d 31 36 b3 66 72**
	Sh_8	10110010 00110101 01101101 00110001 00110110 0**1**110011 01100110 01110010	**B2 35 6D 31 36 73 66 72**
	Sh_9	10110010 00110101 01101101 00110001 00110110 00110011**1**1100110 01110010	**B2 35 6D 31 36 33 E6 72**
	Sh_{10}	10110010 00110101 01101101 00110001 00110110 00110011 0110011**1** 01110010	**B2 35 6D 31 36 33 67 72**
	Sh_{11}	10110010 00110101 01101101 00110001 00110110 00110011 01100110 0111**1**010	**B2 35 6D 31 36 33 66 7A**
	Sh_{12}	10110010 00110101 01101101 00110001 00110110 00110011 01100110 0111001**1**	**B2 35 6D 31 36 33 66 73**
2-Bit	**Sh_{13}**	1**1**11001**1** 00110101 01101101 00110001 00110110 00110011 01100110 01110010	**F3 35 6D 31 36 33 66 72**
	Sh_{14}	1011**1**010 **1**0110101 01101101 00110001 00110110 00110011 01100110 01110010	**BA B5 6D 31 36 33 66 72**
	Sh_{15}	10110**1**10 001101**1**1 01101101 00110001 00110110 00110011 01100110 01110010	**B6 37 6D 31 36 33 66 72**
	Sh_{16}	10110**1**10 00110101 **1**1101101 00110001 00110110 00110011 01100110 01110010	**B6 35 ED 31 36 33 66 72**
	Sh_{17}	10110010 **1**0110101 01101101 **1**0110001 00110110 00110011 01100110 01110010	**B2 B5 6D B1 36 33 66 72**
	Sh_{18}	10110010 00110101 **1**1101101 00110**1**01 00110110 00110011 01100110 01110010	**B2 35 ED 35 36 33 66 72**
	Sh_{19}	10110010 00110101 01101101 00110001 0**1**110110 00110011 0110011**1** 01110010	**B2 35 6D 31 76 33 67 72**
	Sh_{20}	10110010 00110101 01101101 00110001 0011**1**110 00110011 01100110 01110**1**10	**B2 35 6D 31 3E 33 66 76**

This SK can be rephrased in Hexadecimal as SK= (B2 35 6D 31 36 33 66 72), to show the difference with the shares in a comparable manner. This example shows SK having 32 zero bits. Thus, the number of expected shares is estimated to be 528 shares. This number of shares is interestingly big and very useful to be studied. Table 3 mentions the sample of shares here to clarify the concept.

This method gives a good capacity in the number of generated shares. However, it has a vital drawback in all generated shares to be very similar. This is making the probability of guessing SK from shares very high, as can be seen comparing two shares together to guess other shares. The reason behind this security flow is that we changed only one or two 0-bits within the entire 1-block SK sequence for every share, as observed in Table 3. This similarity is completely undesirable in the secret sharing leading to the lowest level of security. The 1-block method is believed to be working against the intended confidentiality property, which requires that there is no information leading to the secret key SK in any part or stage of the secret sharing scheme.

Shares Generated from SK as 2-Blocks

The 1-bit and 2-bits methods are applied on dividing the secret key SK into 2-blocks, i.e., to two equal blocks, where each block contains 32-bits. The 1-bit and 2-bit methods are applied to every block to generate all possible shares. In the 1-bit method, one 0-bit is flipped every 32-bit sequence found in SK block to construct new shares. Similarly, the 2-bit method begins after the 1-bit method to be generating more possible shares. Note that, every time, a different position is to be chosen for zero flipping in each block, in order for the changes to produce new shares from both methods.

The example in Table 4 clarifies the two methods (1-bit and 2-bits) within 2-blocks to generate 'A' shares from SK. Assuming the same SK as in Table 3,

SK = [10110010 00110101 01101101 00110001 00110110 00110011 01100110 01110010] represented in Hex SK= [B2356D3136336672]. The example SK has 32 zero bits possible to generate 16 shares by the 1-bit method, and 120 shares by the 2-bit method, for the 2-blocks. Table 4 shows a sample of the shares generated within this example where the flipped bits have been highlighted.

Table 4. Example listing samples of shares generated assuming SK as 2-blocks.

Binary				Hex	
SK		10110010 00110101 01101101 00110001 00110110 00110011 01100110 01110010		**B2 35 6D 31 36 33 66 72**	
Method	**A**	**Block 1**	**Block 2**	**B2 35 6D 31**	**36 33 66 72**
1-Bit	**Sh_1**	1**0**110010001101010110110100110001	**1**011011000110011011001110010	**F2 35 6D 31**	**B6 33 66 72**
	Sh_2	1011**1**010001101010110110100110001	0**0**110110001100110110011001110010	**BA 35 6D 31**	**76 33 66 72**
	Sh_3	10110010**1**011010101110110100110001	00110110**1**011001101100110011100 10	**B2 B5 6D 31**	**36 B3 66 72**
	Sh_4	101100100**1**101010110110100110001	001101100**1**100110110011001110010	**B2 75 6D 31**	**36 73 66 72**
	Sh_5	1011001000110101**1**110110100110001	0011011000110011**1**100110011100 10	**B2 35 ED 31**	**36 33 E6 72**
	Sh_6	1011001000110101011**1**10100110001	0011011000110011011**0**11001110010	**B2 35 7D 31**	**36 33 76 72**
	Sh_7	1011001000110101011011**1**100110001	0011011000110011011**1**11001110010	**B2 35 6F 31**	**36 33 6E 72**
	Sh_8	1011001000110101011011 01**1**0110001	0011011000110011011001**1**01110010	**B2 35 6D B1**	**36 33 67 72**
	Sh_9	1011001000110101011011010**1**110001	0011011000110011011001 10**1**1110010	**B2 35 6D 71**	**36 33 66 F2**
	Sh_{10}	1011001000110101011011010011**1**001	0011011000110011011001100111**1**010	**B2 35 6D 39**	**36 33 66 7A**
	Sh_{11}	1011001000110101011011010011 0**1**01	0011011000110011011001100111 0**1**10	**B2 35 6D 35**	**36 33 66 76**
	Sh_{12}	1011001000110101011011010011001**1**1	0011011000110011011001100111001**1**	**B2 35 6D 33**	**36 33 66 73**

2-Bits	**Sh_{17}**	1▯11▯01000110101011010100110001	▯▯1101100011001101100110011100 10	**FA 35 6D 31**	**F6 33 66 72**
	Sh_{18}	1▯110▯10001101010110110100110001	▯011▯11000110011011001100111001 0	**F6 35 6D 31**	**BE 33 66 72**
	Sh_{19}	1011▯▯10001101010110110100110001	0▯11▯11000110011011001100111001 0	**BE 35 6D 31**	**7E 33 66 72**
	Sh_{20}	1011▯01▯00110101011011010011000 1	0▯11011▯00110011011001100111001 0	**BB 35 6D 31**	**77 33 66 72**
	Sh_{21}	10110▯1▯00110101011011010011000 1	0011▯11▯00110011011001100111001 0	**B7 35 6D 31**	**3F 33 66 72**
	Sh_{22}	10110▯10▯01101010110110100110001	0011▯110▯01100110110011001110010	**B6 35 6D 31**	**3E 33 66 72**
	Sh_{24}	1011001▯0011▯101011011010011000 1	0011011▯0011▯01101100110011100 10	**B3 3D 6D 31**	**37 3B 66 72**
	Sh_{25}	10110010▯▯1101010110110100110001	00110110▯▯1100110110011001110010	**B2 F5 6D 31**	**36 F3 66 72**
	Sh_{26}	10110010▯01101▯10110110100110001	00110110▯0110▯11011001100111001 0	**B2 B7 6D 31**	**36 B7 66 72**
	Sh_{27}	101100100▯11▯10101101101001100 01	001101100▯11▯01101100110011100 10	**B2 7D 6D 31**	**36 7B 66 72**
	Sh_{28}	101100100▯1101▯10110110100110001	001101100▯110▯11011001100111001 0	**B2 77 6D 31**	**36 77 66 72**
	Sh_{29}	101100100011▯1▯10110110100110001	00110110001▯▯11011001100111001 0	**B2 3F 6D 31**	**36 3F 66 72**
	Sh_{30}	101100100011▯101▯110110100110001	00110110001▯011▯110011001110010	**B2 3D ED 31**	**36 3B E6 72**
	Sh_{33}	101100100011010▯11011▯100110001	00110110001100 11▯110▯11001110010	**B2 35 EF 31**	**36 33 EE 72**
	Sh_{34}	101100100011010101▯11▯100110001	00110110001100110 11▯▯11001110010	**B2 35 7F 31**	**36 33 7E 72**
	Sh_{35}	101100100011010101▯1101▯0110001	00110110001100110 11▯011▯01110010	**B2 35 7D B1**	**36 33 77 72**
	Sh_{36}	101100100011010101101▯1▯0110001	00110110001100110110▯11▯01110010	**B2 35 6F B1**	**36 33 6F 72**
	Sh_{37}	101100100011010101101▯10▯110001	00110110001100110110▯110▯1110010	**B2 35 6F 71**	**36 33 6E F2**
	Sh_{39}	101100100011010101101101011▯001	00110110001100110110011 0111▯010	**B2 35 6D 79**	**36 33 66 FA**
	Sh_{40}	101100100011010101101101001 1▯0▯1	00110110001100110110011001111▯01▯	**B2 35 6D 3B**	**36 33 66 7B**

This method gives a good acceptable capacity of generated shares, but not all shares are useful according to the parallel counting-based technique to reconstruct SK. Therefore, the shares, especially from the 2-bit method, need to be tested before distribution to participants. This 2-blocks method still gives some similarity among generated shares as well as among the shares together, which can be unsecure leading to SK in particular. Thus, 2-blocks strategy gave some probability of guessing SK from shares that may help guess SK, i.e., when compared to more than two shares together, as can be observed in Table 4. This 2-blocks method improved the security compared to the 1-block method, but still noted to be low. This security study will be elaborated later in the comparison and analysis section.

Shares Generated from SK as 4-Blocks

To improve the security of the counting based secret sharing system, we tested implementing the shares generation from two methods (1-bit and 2-bit) on SK partitioned into 4-blocks. The partitioning is aiming to reduce the similarity

between shares and SK although its number of shares is reduced a lot. This 4-blocks model works on dividing SK into four equal blocks of 16-bits. Then, the shares generation 1-bit and 2-bit methods are applied to every block to construct the new shares.

The example in Table 5 clarifies the 1-bit and 2-bit methods applied within 4-blocks shares generation to construct possible shares from SK. Assume the same SK used before represented in Hex as SK= (B2356D3136336672).

Table 5. Example listing samples of shares generated assuming SK as 4-blocks.

		Binary				**Hex**	
SK		10110010 00110101 01101101 00110001 00110110 00110011 01100110 01110010					
Method	**A_{sh}**	**Block 1**	**Block 2**	**Block 3**	**Block 4**	**B2 35 6D 31**	**36 33 66 72**
1-Bit	**Sh_1**	1□110010 00110101	□1101101 00110001	□0110110 00110011	□1100110 01110010	**F2 35 ED 31**	**B6 33 E6 72**
	Sh_2	1011□010 00110101	011□1101 00110001	0□110110 00110011	011□0110 01110010	**BA 35 7D31**	**76 33 76 72**
	Sh_3	10110□10 00110101	011011□1 00110001	0011□110 00110011	0110□110 01110010	**B6 35 6F 31**	**3E 33 6E 72**
	Sh_4	1011001□ 00110101	01101101 □0110001	0011011□ 00110011	0110011□ 01110010	**B3 35 6D B1**	**37 33 67 72**
	Sh_5	10110010 □0110101	01101101 0□110001	00110110 □0110011	01100110 □1110010	**B2 B5 6D 71**	**36 B3 66 F2**
	Sh_6	10110010 0□110101	01101101 0011□001	00110110 0□110011	01100110 0111□010	**B2 75 6D 39**	**36 73 66 7A**
	Sh_7	10110010 0011□101	01101101 00110□01	00110110 0011□011	01100110 01110□10	**B2 3D 6D 35**	**36 3B 66 76**
	Sh_8	10110010 001101□1	01101101 001100□1	00110110 00110□11	01100110 0111001□	**B2 37 6D 33**	**36 37 66 73**
2-Bit	**Sh_9**	1□11□010 00110101	□11□1101 00110001	□□110110 00110011	□11□0110 01110010	**FA 35 FD 31**	**F6 33 F6 72**
	Sh_{10}	1□110□10 00110101	□11011□1 00110001	□011□110 00110011	□110□110 01110010	**F6 35 EF 31**	**BE 33 EE 72**
	Sh_{11}	1□11001□ 00110101	□1101101 □0110001	□011011□ 00110011	□110011□ 01110010	**F3 35 ED B1**	**B7 33 E7 72**
	Sh_{12}	1011□□10 00110101	011□11□1 00110001	0□11□110 00110011	011□□110 01110010	**BE 35 7F 31**	**7E 33 7E 72**
	Sh_{13}	1011□01□ 00110101	011□1101 □0110001	0□11011□ 00110011	011□011□ 01110010	**BB 35 7D B1**	**77 33 77 72**
	Sh_{14}	10110□1□ 00110101	011011□1 □0110001	0011□11□00110011	0110□11□01110010	**B7 35 6F B1**	**3F 33 6F 72**
	Sh_{15}	10110□10 □0110101	011011□1 0□110001	0011□110 □0110011	0110□110 □1110010	**B6 B5 6F 71**	**3E B3 6E F2**
	Sh_{16}	1011001□ □0110101	01101101 □□110001	0011011□ □0110011	0110011□ □1110010	**B3 B5 6D F1**	**37 B3 67 F2**
	Sh_{17}	1011001□ 0□110101	01101101 □011□001	0011011□ 0□110011	0110011□ 0111□010	**B3 75 6D B9**	**36 73 67 7A**
	Sh_{18}	10110010 □□110101	01101101 0□11□001	00110110 □□110011	01100110 □111□010	**B2 F5 6D 79**	**36 F3 66 FA**
	Sh_{19}	10110010 □011□101	01101101 0□110□01	00110110 □011□011	01100110 □1110□10	**B2 BD 6D 75**	**36 BB 66 F6**
	Sh_{20}	10110010 0□11□101	01101101 0011□□01	00110110 0□11□011	01100110 0111□□10	**B2 7D 6D 3D**	**36 7D 66 7E**
	Sh_{21}	10110010 0□1101□1	01101101 0011□0□1	00110110 0□110□11	01100110 0111□01□	**B2 77 6D 3B**	**36 77 66 7B**
	Sh_{22}	10110010 0011□1□1	01101101 00110□□1	00110110 0011□□11	01100110 01110□1□	**B2 3F 6D 37**	**36 3F 66 77**

This 4-blocks example, Table 5, shows that SK of 32 zero bits can be generating only eight shares by the 1-bit method and 28 shares by the 2-bit method. This 4-blocks model provides less number of shares, as low capacity number of shares, compared with the previous two methods, while improving the security as an acceptable tradeoff. It can be noticed from this 4-blocks model that there is some similarity among shares, which may lead intruders to search for SK, as shown in Hex column of Table 5. Therefore, the 4-blocks model can consider the security level as a medium, but is improving among the 2-blocks and 1-block models presented earlier. This security and capacity comparison is detailed later in the comparison and analysis section.

Shares Generated from SK as 8-Blocks

The last proposed enhancement in this shares generation study is the 8-blocks SK model. In this model, we try to reduce the similarity among shares among each other to the best. This experimentation works by dividing SK to 8 equal blocks, where each block contains 8-bits. The 8-blocks model shares generation applies the same philosophy of 1-bit and 2-bit methods but to each block separately. In each block, we chose a different position of one 0-bit or two 0-bits not chosen before, to be flipped for generating a new share.

The following example, shown in Table 6, clarifies this SK 8-blocks model shares generation via the 1-bit and 2-bit methods. Assume the same SK used before represented in Hex as SK= (B2356D3136336672). Through this 8-blocks example, as in Table 6, the number of shares generated by the 1-bit method and 2-bit method is only four shares and six shares, respectively. This model gives high security, but low number of shares compared to all previous methods. It is noted that it is very difficult to guess SK from shares, i.e., when some illegal shares are considered together. Interestingly, we found the similarity among shares almost nonexistent, as observed in the Hex columns of Table 6. This method's main drawback is the very low capacity in generated shares, as expected from SK to be much lower than all the previous models. Therefore, we can consider the security level in this model as high but at the expense of capacity, which we will study more later in the comparison and analysis section.

Table 6. Example listing shares generated assuming SK as 8-blocks.

Binary										Hex							
SK		10110010 00110101 01101101 00110001 00110110 00110011 01100110 01110010								B2 35 6D 31 36 33 66 72							
Method	**A**	**Block 1**	**Block 2**	**Block 3**	**Block 4**	**Block 5**	**Block 6**	**Block 7**	**Block 8**	**B2**	**35**	**6D**	**31**	**36**	**33**	**66**	**72**
1-Bit	**Sh_1**	1▯110010	▯0110101	▯1101101	▯0110001	▯0110110	▯0110011	▯1100110	▯1110010	**F2**	**B5**	**ED**	**B1**	**B6**	**B3**	**E6**	**F2**
	Sh_2	1011▯010	0▯110101	011▯1101	0▯110001	0▯110110	0▯110011	011▯0110	0111▯010	**BA**	**75**	**7D**	**71**	**76**	**73**	**76**	**7A**
	Sh_3	10110▯10	0011▯101	011011▯1	0011▯001	0011▯110	0011▯011	0110▯110	01110▯10	**B6**	**3D**	**6F**	**39**	**3E**	**3B**	**6E**	**76**
	Sh_4	1011001▯	001101▯1	01101101	00110▯01	0011011▯	00110▯11	0110011▯	0111001▯	**B3**	**37**	**6D**	**35**	**37**	**37**	**67**	**73**
2-Bit	**Sh_5**	1▯111▯010	▯0110101	▯11▯1101	▯0110001	▯0110110	▯0110011	▯111▯0110	▯1111▯010	**FA**	**F5**	**FD**	**F1**	**F6**	**F3**	**F6**	**FA**
	Sh_6	1▯1101▯10	▯011▯101	▯11011▯1	▯011▯001	▯011▯110	▯011▯011	▯110▯110	▯1110▯10	**F6**	**BD**	**EF**	**B9**	**BE**	**BB**	**EE**	**F6**
	Sh_7	1▯11001▯	▯01101▯1	011▯11▯1	▯0110▯01	▯011011▯	▯0110▯11	▯110011▯	▯111001▯	**F3**	**B7**	**7F**	**B5**	**B7**	**B7**	**E7**	**F3**
	Sh_8	1011▯▯10	0▯11▯101	01101101	▯01100▯1	0▯11▯110	0▯11▯011	011▯▯110	0111▯▯10	**BE**	**7D**	**6D**	**B3**	**7E**	**7B**	**7E**	**7E**
	Sh_9	1011▯01▯	0▯1101▯1	01101101	0▯11▯001	0▯11011▯	0▯110▯11	011▯011▯	0111▯01▯	**BB**	**77**	**6D**	**79**	**77**	**77**	**77**	**7B**
	Sh_{10}	10110▯1▯	0011▯1▯1	01101101	0▯110▯01	0011▯11▯	0011▯▯11	0110▯11▯	01110▯1▯	**B7**	**3F**	**6D**	**75**	**3F**	**3F**	**6F**	**77**

Secret Key Retrieval from M-Blocks Shares

The secret key retrieval mechanism from specific shares in our new approach follows the same the basic original counting-based secret sharing approach (Gutub et al., 2017). All 'A' shares generated from SK by the 1-bit and 2-bit methods within M-blocks are combined based on the threshold in a parallel counting manner. The selected 'n' shares are the number of participants in the secret sharing system. The shares of 1-bit method are all fine. However, the shares from the 2-bit method need to be tested to make sure of their validity before distributing to the authorized contributors. When the need is to reconstruct the original secret key SK, the application should have determined a value of k out of n ($n \geq$ number of valid shares $\geq k$) assumed as the threshold of existing accurate shares, considering matching for SK redevelopment. The k shares are entered to the system and considered in parallel via the counting-based rules for parallel bits to be calculated. If the counting result out of the assigned bits' grouping for one column comes out as quantity (k) or larger, then the remarking related output location is given bit one; otherwise, the bit is remarked zero, and so on. These remarked outcome bits are equated with starting correct SK to confirm authentication. This process is not affected by the M-blocks modeling and can be applied as briefed earlier in Fig. 3.

Application of 8-Blocks Model Possibilities

This section will focus on 8-blocks SK model. It will present dissimilar scenarios for operation of shares to simplify SK recovery mechanism and potential application trials to be discussed. Assume the same secret key SK existing in this paper, SK= (B2 35 6D 31 36 33 66 72). The model used 'A' shares generated by the 1-bit and 2-bit methods within 8-blocks model as presented in Table 6 before. Assuming the number of selected shares is chosen for users with n=8 (Table 7). The specific shares distributed have been tested to be valid and applicable to recover SK secret key with threshold k=5.

Table 7. Valid shares selected from 8-blocks model distributed to 8 participants.

n_{sh}	Binary share	Hex share
Sh_1	11110010101101011110110110110001101101101011001111100110111 10010	**F2 B5 ED B1 B6 B3 E6 F2**
Sh_2	10111010011101010111110101110001011101100111001101110110011 11010	**BA 75 7D 71 76 73 76 7A**
Sh_3	10110110001111010110111100111001001111100011101101101110011 10110	**B6 3D 6F 39 3E 3B 6E 76**
Sh_4	10110011001101110110110100110101001101110011011101100111011 10011	**B3 37 6D 35 37 37 67 73**
Sh_5	11111010111101011111110111110001111101101111001111110110111 11010	**FA F5 FD F1 F6 F3 F6 FA**
Sh_7	11110011101101110111111110110101101101111011011111100111111 10011	**F3 B7 7F B5 B7 B7 E7 F3**
Sh_9	10111011011101110110110101111001011101110111011101110111011 11011	**BB 77 6D 79 77 77 77 7B**
Sh_{10}	10110111001111110110110101110101001111110011111101101111011 10111	**B7 3F 6D 75 3F 3F 6F 77**

Case 1: Situation of Combining Shares = k

In this case, let k=5 be the number of shares. This scenario is well-thought as valid. Thus, the shares are joint to reconstruct SK, as in the following example:

Sh_1 :	11110010101101011110110110110001101101101011001111100110111100 10	**>> F2 B5 ED B1 B6 B3 E6 F2**
Sh_3 :	10110110001111010110111100111001001111100011101101101110011101 10	**>> B6 3D 6F 39 3E 3B 6E 76**
Sh_4 :	10110011001101110110110100110101001101110011011101101100111011 0011	**>> B3 37 6D 35 37 37 67 73**
Sh_5 :	11111010111101011111110111110001111101101111001111110110111110 10	**>> FA F5 FD F1 F6 F3 F6 FA**
Sh_{10} :	10110111001111110110110101110101001111110011111101101111011101 11	**>> 96 3D 6B 39 3D 3B 7E 76**

K :	52551252215525252551551522551205215525522155225525512552255512 52	**>> Counting Results**
SK =	10110010001101010110110100110001001101100011001101100110011001 10010	**>> B2 35 6D 31 36 33 66 72**

The recovered output of the combination process equals Hex value B2 35 6D 31 36 33 66 72, which is equal to the secret key SK in a proven correct situation.

Case 2: Situation of Combining Shares > k.

In this case, assume k=5 defining number of shares > k. This scenario should be measured correctly, although the number of shares is greater than k. The shares reconstructed with this situation proved the counting result of bits as to be larger than or equal to k, as in the example

Sh_1 :	11110010101101011110110110110001101101101011001111100110111100 10	**>> F2 B5 ED B1 B6 B3 E6 F2**
Sh_3 :	10110110001111010110111100111001001111100011101101101110011101 10	**>> B6 3D 6F 39 3E 3B 6E 76**
Sh_4 :	10110011001101110110110100110101001101110011011101101100111011 0011	**>> B3 37 6D 35 37 37 67 73**
Sh_5 :	11111010111101011111110111110001111101101111001111110110111110 10	**>> FA F5 FD F1 F6 F3 F6 FA**
Sh_9 :	10111011011101110110110101111001011101110111011101110111011101 11011	**>> BB 77 6D 79 77 77 77 7B**
Sh_{10} :	10110111001111110110110101110101001111110011111101101111011101 11	**>> 96 3D 6B 39 3D 3B 7E 76**

K :	62662263226626362661661623662206226626632266236626622663263263 26263	**>> Counting Results**
SK =	10110010001101010110110100110001001101100011001101100110011001 10010	**>> B2 35 6D 31 36 33 66 72**

The number of shares is found larger than k. So, as the counting sum outcome of bits in every column ≥ k, the counting result implies giving one; otherwise, it implies giving a zero, as placed in the testing SK location. Thus, The Hex output of the combination process is B2 35 6D 31 36 33 66 72, which is equal to the secret key SK as proven correct.

Case 3: Situation of Combining Shares < k.

In this case, assume k=5 and number of shares < k; the system should stop accessibility making it unable to retrieve SK. The condition of number of shares found to be less than k is shown with only 4 shares, as follows:

Sh_1 : 11110010101101011110110110110001101101101011001111100110111100 10 **>> F2 B5 ED B1 B6 B3 E6 F2**

Sh_3 : 10110110001111010110111100111001001111100011101101101110011101 10 **>> B6 3D 6F 39 3E 3B 6E 76**

Sh_4 : 10110011001101110110110100110101001101110011011101100111011001 11 **>> B3 37 6D 35 37 37 67 73**

Sh_9 : 10111011011101110110110101111001011101110111011101110111011101 11 **>> BB 77 6D 79 77 77 77 7B**

K : 41441142114414241440441411442104114414421144124414411442144411 42 **>> Counting Results**

SK = 00000000000000000000000000000000000000000000000000000000000000 00 **>> 00 00 00 00 00 00 00 00**

In this case, the number of shares is less than k. Thus, the outcome of the combination process is 0000000000000000 ≠ SK= B2 35 6D 31 36 33 66 72. Hence, it cannot recover the secret key.

Case 4: Situation of involving intruder False Share

Assume that a trespasser comes in with a false share in the given example of Case 1, i.e., k=5, and number of shares = k, but this single share (or can be more) is false, i.e., injected by hacker, as in the following example:

Sh_1 : 11110010101101011110110110110001101101101011001111100110111100 10 **>> F2 B5 ED B1 B6 B3 E6 F2**

Sh_3 : 10110110001111010110111100111001001111100011101101101110011101 10 **>> B6 3D 6F 39 3E 3B 6E 76**

Sh_4 : 10110011001101110110110100110101001101110011011101100111011001 11 **>> B3 37 6D 35 37 37 67 73**

Sh_9 : 10111011011101110110110101111001011101110111011101110111011101 11 **>> BB 77 6D 79 77 77 77 7B**

FSh: 10010110001111010110101100111001001111010011101101111110011101 10 **>> 96 3D 6B 39 3D 3B 7E 76**

K : 51451252115525251550542511553105115525431155225515522552155512 52 **>> Counting Results**

SK = 10010010001101010110100100110001001101000011001101100110011101 10 **>> 92 35 69 31 34 33 66 72**

This case is considered inacceptable, where all shares are legal but the false one marked as Fsh. Thus, the output of the combination process is different such as 92 35 69 31 34 33 66 72 ≠

SK= B2 35 6D 31 36 33 66 72, confirming the model security correctness.

Case 5: Situation of involving intruder several False Shares

Assume a similar case, to Case 4 above, of intruder inserting false shares, but with more than one false share, namely, FSh_1 and FSh_2, as in the following example:

Sh_1 :	1111001010110101111011011000110110110101100111110011011110010	**>> F2 B5 ED B1 B6 B3 E6 F2**
Sh_3 :	1011011000111101011011110011100100111110001110110110111001110110	**>> B6 3D 6F 39 3E 3B 6E 76**
Sh_4 :	1011001100110111011011010011010100110111001101110110011101110011	**>> B3 37 6D 35 37 37 67 73**
FSh_1:	1011011000110111010111010101010101010110011101110111010001110100	**>> B6 37 5D 55 56 77 74 74**
FSh_2:	1001010000111101011010110001100100111101001110110111110001110110	**>> 94 3D 6B 19 3D 3B 7C 76**

K :	5145034110552525154154251135220511542542115522551552253115550342	**>> Counting Results**
SK =	1001000000110101010010010001000100100100001100110110010001110000	**>> 90 35 49 11 24 33 64 70**

The two false shares when combined to the valid shares give invalid counting-based results. Thus, the Hex outcome of the combination process is 90 35 49 11 24 33 64 70 ≠ SK, which verifies the scheme security strength.

COMPARISON AND RESULTS ANALYSIS

The proposed new shares generation via M-blocks secret key partitioning for counting-based secret sharing has been implemented and compared involving all studied models, namely, 1-block, 2-blocks, 4-blocks, and 8-blocks models, considering the 1-block model as the original basic counting-based secret sharing scheme but modified to be for 64-bits in size, in order to provide fair research and analysis. The security approximation is gained from considering the average similarity rate found between SK secret key aligned with the generated shares. The study is determining the level of security for these M-blocks models suggesting the best method based on less similarity among the shares and among SK secret key. So, the work depended on the same secret key SK (SK= B2 35 6D 31 36 33 66 72) used fairly in all examples to generate the different shares. The security comparison is built on similarity rate (S) for digits of any share calculated among all other shares. This S rate is generated by calculating the number of times frequency (F) for a digit in the same location found in all shares. Then, the analysis is calculating percentage frequency for each share independently using formula: .

The average similarity rate (V_{sh}) among all shares is computed for every M-block model. Likewise, the similarity rate (S) is calculated pairing SK digits with shares digits using the number of frequency (F) per digit in the same location, i.e., for shares with the same digit location in SK. For example, the similarity ratio (S) for the share assuming model of Ash=1 can be calculated as: , that is used for calculating the average similarity rate (V_{sk}) between the shares as listed in Table 8.

Table 8. Average similarity rate among shares compared to SK.

1-Bit & 2-Bit Methods	A-share	Average of Similarity Rate	
		V_{sh}	V_{sk}
Original 1-Block (Entire Sequence)	528	78.43 %	88.10
Within 2-Blocks	136	62.13 %	77.34
Within 4-blocks	36	41.84 %	58.85
Within 8-Blocks	10	28.75 %	34.38

Consider the results (Table 8) of the original 1-block model gained from 1-bit and 2-bit methods calculated assuming the SK size of 64-bits. The similarity rate (Vsh) is found to be 78.43% above the average, while the average of similarity rate (Vsk) in one share with SK is found higher around 88.10%, as shown in Fig. 6. It is to be noted that this 1-block model is providing around five times the number of shares of all the other models as a normal tradeoff between security and capacity, which needs to be optimized. This increase in the shares capacity (A-shares) is due to the increase of SK size to be 64-bit involving 50% zeros, allowing the numbers of generated shares to be 528 shares, as shown in Fig. 6. This increase in shares capacity (number of shares) is reflecting the expense of the security level. Consequently, this method is not preferred to be used in the counting-based secret sharing scheme CBSSS, because of its lowest security, i.e., giving the high rate of similarity between shares and between the one share with the secret key.

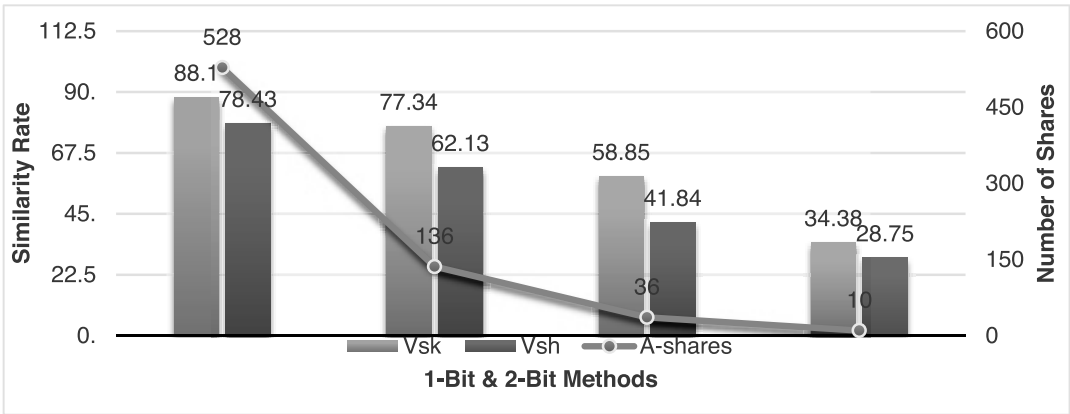

Figure 6. Similarity rate among shares comparing the M-block models.

The 2-blocks model improved the previous 1-block original scheme enhancing the security. The results appeared simple improvement reducing the average of similarity rate (Vsh) to be 62.13% and the average similarity rate (Vsk) decreased to 77.34%. The capacity of shares generating by this method is considered lower than the1-block model, but can be acceptable based on the user application, as observed in Fig. 6. In fact, the security level of this 2-blocks method can be observed to be still low as will be explained in the coming models.

The 4-blocks model shows clear improvement compared to the previous two models, 1-block and 2-blocks models. Its similarity between shares is found to reduce the rate (Vsh) to 41.84 % and the average similarity rate (Vsk) among SK reduced to 58.85%, as observed in Fig. 6. The capacity of shares decreased a lot in this model, which is statistically due to increase in the number of flipped zeros, which approximately changed the entire SK sequence.

The most secure model in this study is the 8-blocks model. It gave the best results among the rates of the previous models, as observed in Fig. 6. This 8-blocks model showed that the average of similarity rate (Vsh) and (Vsk) reduced to 28.75% and 34.38%, respectively. In fact, as expected, this 8-blocks model gives the lowest capacity in number of shares making its degradation in its applicability. Therefore, it is preferred to choose the appropriate M-block model based on the application need, i.e., to generate the sufficient number of shares with recommended security for the realistic CBSSS.

To prove the study further, for generalizing the contributions, we tested the M-block CBSSS on many other random samples of SK, as listed in Table 9. Interestingly, the similarity rates (Vsh) and (Vsk) varied relying on the number of zeros within the RNG generated secret key, as well as the distribution mechanism of these zeros within the secret key sequence, as visualized in Fig. 7 and Fig. 8. This comprehensive testing permitted stating the contribution of the 8-blocks model to the optimal method to generate shares recommended for running secure counting-based secret sharing scheme CBSSS.

Table 9. Average similarity rate of shares comparing different SKs applied to M-block models.

N	Hex SK	Number of Zeros	Original 1-Block		Within 2-Blocks		Within 4-blocks		Within 8-Blocks	
			V_{sh}	V_{sk}	V_{sh}	V_{sk}	V_{sh}	V_{sk}	V_{sh}	V_{sk}
SK_1	15CBB4CAC20FD5E4	32	78.73	88.16	63.09	77.57	45.65	61.28	34.25	41.25
SK_2	854B78391A5F4DA3	32	78.61	88.14	64.49	78.77	45.61	61.46	33.38	40
SK_3	9A292DF2C52B71E9	31	78.7	88.16	64.55	78.75	47.39	63.39	35.59	37.5
SK_4	ED8D3C2B25C3A749	31	78.57	88.13	65.82	79.81	46.84	62.86	35.59	41.67
SK_5	11E69633E9CC53B0	33	78.47	88.11	62.37	77.34	44.12	60.76	30.13	38.13
SK_6	70D6872AD85DC6A5	32	78.61	88.14	64.48	78.72	45.35	61.63	31.56	39.38
SK_7	E05E7C3FCDA5279C	28	78.89	88.19	69.79	82.46	52.05	67.32	35.24	38.54
SK_8	179BF475C4D6F891	29	78.95	88.2	68.31	81.2	50.67	65.89	36.11	38.54
SK_9	B1E3258E374F319E	30	78.78	88.17	66.44	80.05	49.98	65.36	35.42	36.46
SK_{10}	4BEA6A76C0F2D525	31	78.76	88.17	66.46	79.95	47.51	63.39	34.9	37.5
SK_{11}	D5C1B2C4261A9601	38	78.3	88.08	62.69	77.47	42.46	58.78	29.5	35
SK_{12}	CC02DD9120213D41	40	78.32	88.09	64.1	77.77	45.57	61.68	34.53	44.58
SK_{13}	ED509BC4962D794A	32	78.61	88.14	62.68	77.48	43.21	59.38	34.94	43.75
SK_{14}	2D68073619A89E36	35	78.43	88.11	62.64	77.42	45.49	61.11	30.19	35.63
SK_{15}	C48733170FC20E2E	35	78.68	88.15	63.4	77.51	45.66	59.38	32.13	35
SK_{16}	D407141615A0C1DC	39	78.36	88.09	63.35	77.56	44.8	59.29	30.94	36.88
SK_{17}	D8ADC09623105CE6	36	78.45	88.11	63	77.58	48.35	64.19	30.44	38.13
SK_{18}	D2617CE1CF8C88E2	33	78.65	88.15	62.85	77.44	48.4	63.89	35.88	43.13
SK_{19}	559AA8174F5E7E92	30	78.72	88.16	67.73	81.06	49.3	65.18	34.55	40.63
SK_{20}	24032040A016808C	49	78.02	88.03	64.44	77.78	44.92	59.84	22.68	31.85
SK_{21}	352600002840EBA2	45	78.22	88.07	63.23	77.64	47.38	62.82	30.92	40
SK_{22}	0819005080400321	52	77.97	88.02	65.18	78	46.96	61.43	21.64	30.06
SK_{23}	7108E8604000E92A	44	78.29	88.08	64.35	77.77	44.03	59.94	31.44	41.67
SK_{24}	528800B138D07000	46	78.24	88.07	63.29	77.62	44.19	60.26	27.61	35.42
SK_{25}	6C18208801521202	48	78	88.03	63.96	77.73	44.63	60	24.39	35.12

COMPARISONS WITH OTHER CBSSS METHODS

In this section, we will present a comparison of our proposed M-Block partitioning work of this paper with other CBSSS methods that have been presented as modifications serving the counting-based secret sharing scheme. The comparison is considering research of Taghrid Al-Khodaidi (2019) as well as Maimoona Al-Ghamdi (2018) in a separate manner raising related work different views. First, the comparison focused on Taghrid Al-Khodaidi (2019)

study observing the interesting different ways to generate the secret key, in order to get the reliable SK from the works. The second comparison focused on Maimoona Al-Ghamdi (2018) proposal in terms of shares generation schemes seeking optimized manners.

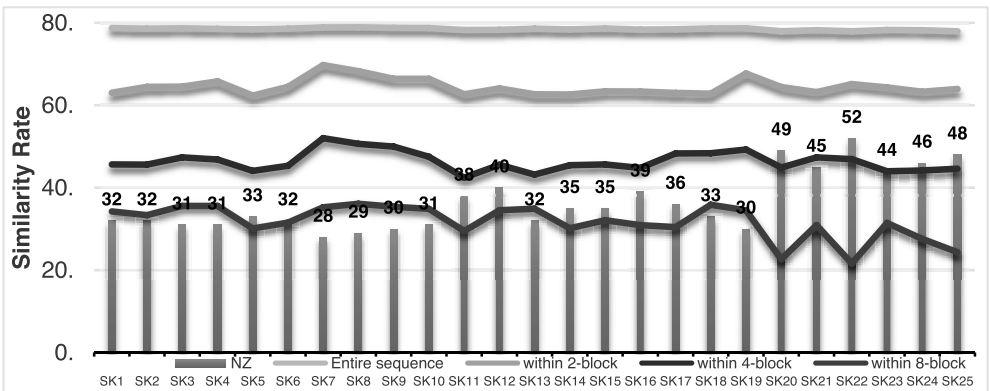

Figure 7. Random samples of SKs similarity rate among all shares for all M-block models.

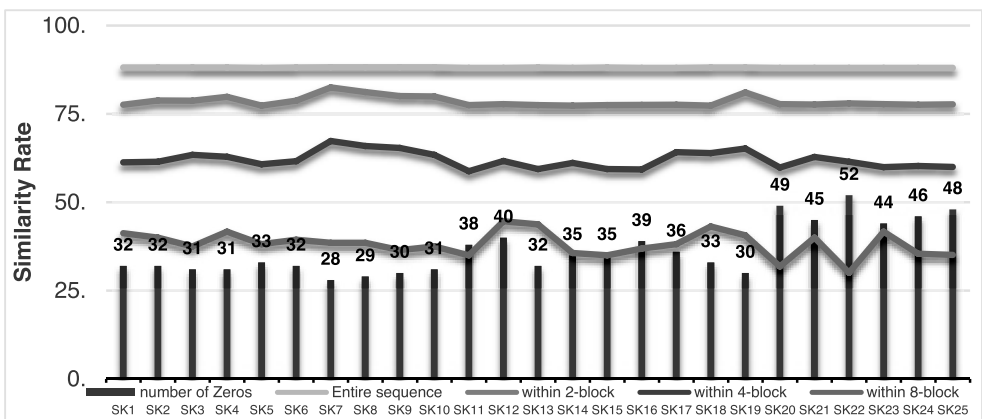

Figure 8. Random sample of SKs similarity among one share with SK for all M-block models.

As briefed earlier, Taghrid Al-Khodaidi (2019) work proposed the methods to produce seven secret keys SKs. Then, the system compares between all seven secret keys SKs to find the best secret key SK to use. Our work considered the comparison of SK to be performed with (Al-Khodaidi et al., 2019) seven random secret keys generated with the same RNG tool used in our work to compare them. Therefore, this study comparison will be based on the statistical tests from NIST to analyze the results. Note that NIST presents several standard statistical tests to measure the randomness of the secret key sequences (Rukhin et al., 2001). We selected some of these NIST tests that may be useful in comparing the reliability level for the secret key as generated by the proposed methods in counting-based secret sharing schemes. This research used Frequency test and Frequency test within a block as two tests from NIST 800-22 tests (Rukhin et al., 2001) to evaluate the reliability of the secret key to calculate the p-value (AlQurashi et al. 2018). The p-value specifies the randomness indicating more applicable result of the secret key. As studied in AlQurashi et al. (2018), if the p-value is less than 0.01, the secret key is not acceptable and is considered nonrandom; otherwise, the secret key is randomly fine.

In this comparison study, we have determined the evaluation standard estimation that relates between p-value from frequency test and p-value from frequency test within a block, as a combined figure of merit value representing a preference estimation, namely, secret key weight. The aim of this comparison study is to analyze the best reliable

secret key from both works tested between each other. The value of the secret key weight assumes both frequency and frequency within a block tests having the same priority weight, as presented in AT principle in the literature (Gutub, 2006) for reasonable efficiency estimation, as AT (area × speed). Therefore, we compute the secret key weight by similar combination equation, Secret Key weight = p-value of Frequency × p-value of Frequency with a block; and the results are listed in Table 10.

Table 10. Reliability of secret keys in our work compared to secret keys in Taghreed's work.

Our M-blocks partitioning work					Taghreed's work (2019)				
Secret Key	**Number of zeros**	**Frequency (Monobit)**	**Frequency with a Block**	**SK wieght**	**Secret Key**	**Number of zeros**	**Frequency (Monobit)**	**Frequency with a Block**	**SK wieght**
SK_1	32	0.802587	0.992708	0.79673	SK_1	42	0.012419	0.43347	0.005383
SK_2	31	1	0.746837	0.746837	SK_2	37	0.2113	0.961731	0.203214
SK_3	32	0.802587	0.891292	0.715339	SK_3	42	0.0455	0.647232	0.029449
SK_4	33	0.317311	0.84799	0.269076	SK_4	42	0.012419	0.151204	0.001878
SK_5	30	0.617075	0.746837	0.460854	SK_5	40	0.024449	0.377154	0.009221
SK_6	35	0.453255	0.992708	0.44995	SK_6	41	0.0455	0.636031	0.028939
SK_7	33	0.802587	0.580338	0.465772	SK_7	42	0.012419	0.33393	0.004147

Considering the comparison results listed in Table 10, all the samples selected are secret keys passing the two tests (frequency test and frequency test with a block) in both works, i.e., this work and the acceptable best results of Taghrid Al-Khodaidi work (2019). Note that they both gave random sequence showing observed interesting differences. It is to be remarked that all the seven experiments for the best secret keys generated by Al-Khodaidi (2019) models show lower level secret key weight compared to the secret keys weight generated by our work RNG of M-blocks partitioning presented here. This identification shows high reliability of the secret keys to be valid from our work as preferred recommended to be used in counting-based secret sharing scheme, as shown clearly in Fig. 9.

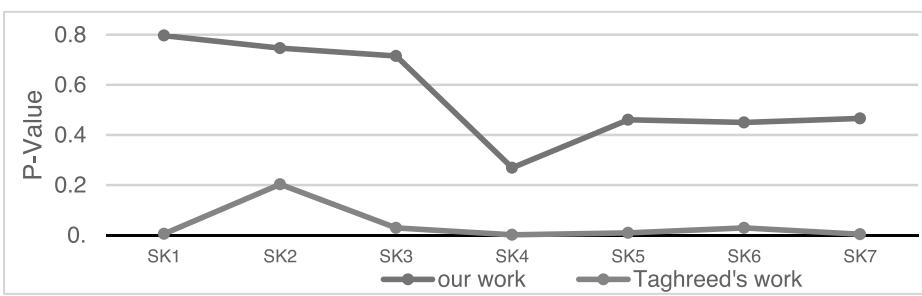

Figure 9. Reliability of secret key in our work compared with Taghreed's work.

Consider the related work presented by Maimoona Al-Ghamdi (2018) that suggested to similarly enhance security and efficiency of CBSSS but affecting the shares generation process with different manner. The comparison study enabled verification tool by choosing valid secret key to determine the shares generation models in a fair research platform. In fact, this comparison was performed involving all studied models from (Al-Ghamdi et al., 2018) in both works presented in relation to the four proposed models. The comparison considered our M-block generation of shares

via 1-block, 2-blocks, 4-blocks, and 8-blocks, to be similar to the three proposed models (Basic, New, and Merged) of the work of Al-Ghamdi et al. (2018). We discussed the security levels analysis of the different proposed models in both works by computing the average similarity rate (Vsh) among all shares generated from the models, as well as the average similarity rate (Vsk) between the share within the secret key as mentioned in this M-block partitions CBSSS paper. In addition, this comparison includes computing the number of shares generated in each model. Table 11 shows that these comparisons depended on their outcomes existing from our work, as well as the work of Al-Ghamdi et al. (2018). We used three experimentations of the secret key, which suits each method in a fair comparison manner.

Table 11. Security rate compared to the work of Al-Ghamdi et al. (2018) models.

Work	Experiment	Model	Number of shares	V_{sh}	V_{sk}
Others (Al-Ghamdi et al., 2018)	SK_1	Basic Shares Generation	820	78.28	88.08
	SK_2	New shares generation	666	78.30	88.08
	SK_3	Merged Shares Generation	1056	77.96	88.10
Our work	SK_1	Original 1-Block	528	78.43	88.10
		Within 2-Blocks	136	62.13	77.34
		Within 4-blocks	36	41.84	58.85
		Within 8-Blocks	10	28.75	34.38

Consider the results of Table 11 showing all shares generation models proposed by Maimoona>s work (Al-Ghamdi et al., 2018) as well as our M-block partitioning proposal in a fair comparison listing. The similarity rate (Vsh) of the other works appeared to raise above the average of all shares with a percentage more than 77%, as in Fig. 10. The average of similarity rate (Vsk) among the shares of SK raised around 88 %, as also shown clearly in Fig. 11. Note that the 1-block model representing the original shares generation method in our work showed the same range of similarity rate (Vsh) and (Vsk) to the work of Al-Ghamdi et al. (2018), as detailed visually in Fig. 10 and Fig. 11. In fact, these percentages are considered very high indicating low security level compared to all other M-block models, i.e., 2-blocks, 4-blocks, and 8-blocks of our proposed work with similarity rate (Vsh) of percentages values 62 %, 41%, and 28%, respectively. Remark the average similarity rate (Vsk) reduction of percentages to all models as shown in Fig. 11 confirming the fact. Consequently, this low similarity rate observed related to our M-block partitioning scheme of both (Vsh) and (Vsk) leads to higher level of security. This can be considered as an advantage of the shares generation models presented in our work as giving higher security level of shares compared to all models of the work of Al-Ghamdi et al. (2018).

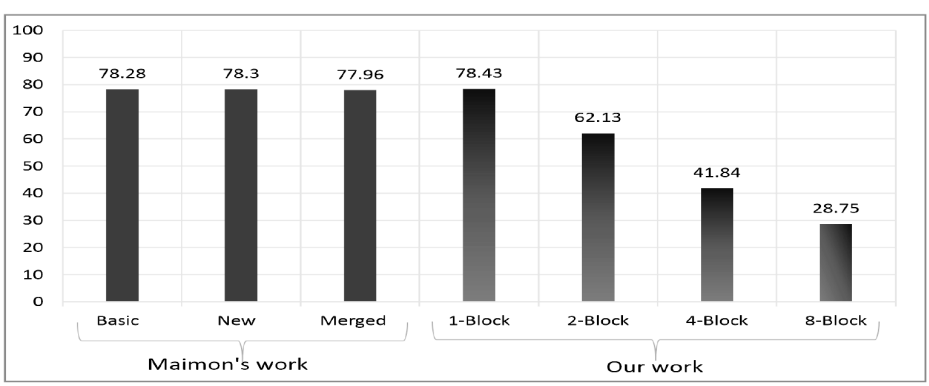

Figure 10. Similarity rate among shares in both works.

On the other hand, all presented models of generating shares in the work of Al-Ghamdi et al. (2018) gave a large number of shares compared to our proposed M-block partitioning models in our work. This fact is found clear considering the merged shares generation model in Maimoona's work (Al-Ghamdi et al., 2018). As expected, the number of generated shares in each model of our work reduced by rising the security level as shown in Fig. 12, which made clear tradeoff price to be paid. This tradeoff issue is not the case with Maimoona Al-Ghamdi (2018) work showing that increasing number of shares found appropriate CBSSS strategy whenever applications need to have very large number of shares.

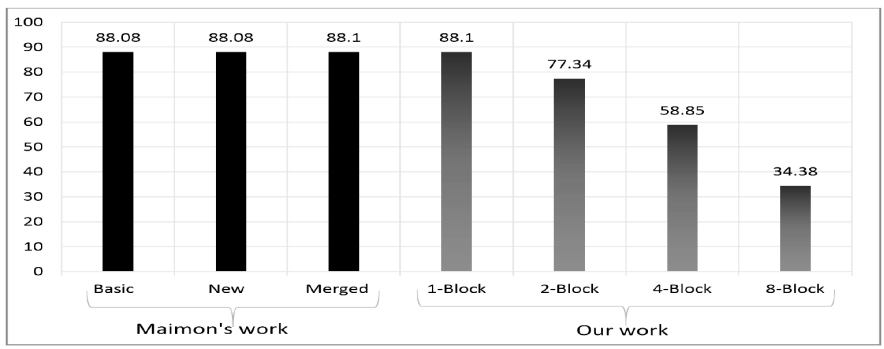

Figure 11. Similarity rate among one shares with SK in both works.

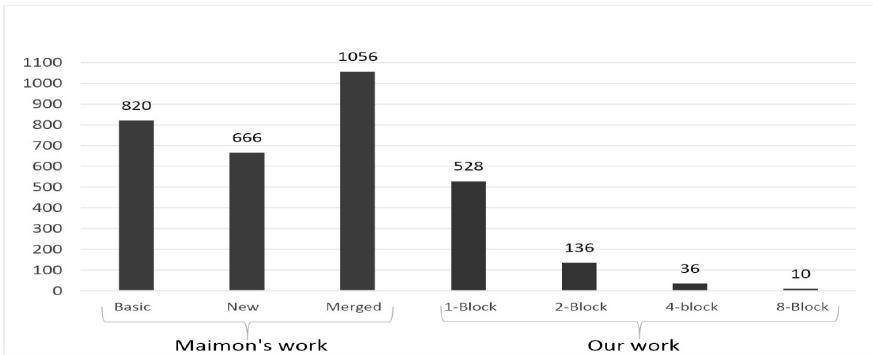

Figure 12. The number of generated shares in both works.

CONCLUSION

This work has presented secret key SK M-block partitioning models for shares generation of counting-based secret sharing for the objective of increasing security. The implementation designed the system increasing its size of SK to 64-bits as the minimum required password in real-life applications. The research considered several M-blocks models, namely, the 1-block original model, 2-blocks, 4-blocks, and 8-blocks models. All models shares generations have been performed via the 1-bit and 2-bit methods applied within each block, i.e., to generate the shares. The study detailed the models' pros and cons testing all possibilities and further generalized the remarks assuming openly tested secret keys pretending real-life scenarios.

The research noticed clear effect of each model on the security level within generated shares from SK. We calculated the averages of the similarity rate among all shares generated by the different models as well as the average similarity rates of shares among SK providing interesting security level remarks. The study proved determining the level of security to be in relation to the application since it clarified the tradeoff fact between security and number of shares (pretending capacity).

The results showed the rise in the average percentage of the similarity rate among all shares compared to each other expressing the rates by the 1-block original, 2-blocks, 4-blocks, and 8-blocks models, as average percentages of 78%, 66%, 47%, and 28%, respectively. Similarly, the study also considered the average percentage rate of the similarity rate among one share with SK as the rates of the 1-block original, 2-blocks, 4-blocks, and 8-blocks models, as average percentages of 88%, 80%, 62%, and 37%, respectively. These average similarity rates have been consistently decreasing as the number of blocks increases indicating that higher number of blocks provides higher security. However, as the security increases, the number of shares capacity decreases showing a real tradeoff optimization challenge to be addressed as future work. It is clearly noted that whenever we are increasing the division partitions within the secret key as more numbers of blocks, the security increases with the expense taken from the number of shares as limiting the number of application users. This problem of increasing the security level in shares generation methods opened the issue of the lower capacity to produce shares that need to be further researched.

The study concluded that the 8-blocks model is providing the highest security to generate shares running the counting-based secret sharing scheme, which can be useful and recommended for applications with limited number of users not requiring a lot of shares, such as nuclear missile launch control, opening the vault in central banks, considerable medical agreement, and sensitive encryption keys. Further future research can recommend varying the block sizes as well as increasing the size of SK to 128-bit and analyzing its effect on security and capacity. Also, we can suggest other methods to increase vagueness on shares generation in counting-based secret sharing pretending to increase the security within more numbers of users.

REFERENCES

Alassaf, N., Gutub, A., Parah, S. & Al-Ghamdi, M. 2018. Enhancing Speed of SIMON: A Light-Weight-Cryptographic Algorithm for IoT Applications. Multimedia Tools and Applications, ISSN: 1380-7501, DOI: 10.1007/s11042-018-6801-z.

Alsaidi, A., Al-lehaibi, K., Alzahrani, H., Al-Ghamdi, M. & Gutub, A. 2018. Compression Multi-Level Crypto Stego Security of Texts Utilizing Colored Email Forwarding. Journal of Computer Science & Computational Mathematics (JCSCM), Vol. 8, No. 3, pp. 33-42, DOI: 10.20967/jcscm.2018.03.002, Published by Science & Knowledge Research Society.

Alaseri, K. & Gutub, A. 2018. Merging Secret Sharing within Arabic Text Steganography for Practical Retrieval. IJRDO - Journal of Computer Science and Engineering, ISSN: 2456-1843, Vol. 4, No. 9.

AlQurashi, A. & Gutub, A. 2018. Reliable Secret Key Generation For Counting-Based Secret Sharing. Journal of Computer Science & Computational Mathematics, Vol. 8, No. 4.

Al-Juaid, N., Gutub, A. & Khan, E. 2018. Enhancing PC Data Security via Combining RSA Cryptography and Video Based Steganography. Journal of Information Security and Cybercrimes Research (JISCR), Vol. 1, No. 1, pp. 8-18, Published by Naif Arab University for Security Sciences (NAUSS).

Al-Ghamdi, M., Al-Ghamdi, M. & Gutub, A. 2018. Security Enhancement of Shares Generation Process for Multimedia Counting-Based Secret-Sharing Technique. Multimedia Tools and Applications, ISSN: 1380-7501, DOI: 10.1007/s11042-018-6977-2.

Al-Khodaidi, T. & Gutub, A. 2019. Scalable Shares Generation to Increase Participants of Counting-Based Secret Sharing Technique. International Journal of Information and Computer Security, ISSN 1744-1765, In Press.

Bai, Li, & XuKai, Zou 2009. A proactive secret sharing scheme in matrix projection method. International Journal of Security and Networks, Vol. 4, No. 4, pp. 201-209.

Beimel, A., Tassa, T. & Weinreb, E. 2005. Characterizing ideal weighted threshold secret sharing. Theory of Cryptography Conference, Springer, pp. 600-619.

Blakley, G.R. 1979. Safeguarding cryptographic keys. Proc. of AFIPS National Computer Conference, Vol. 48, pp. 313-317.

Binu, V.P. & Sreekumar, A. 2016. Secret Sharing Schemes with Extended Capabilities and Applications. Diss. Cochin University of Science and Technology.

Castiglione, Arcangelo, Alfredo, De Santis, Barbara & Masucci, 2014. Hierarchical and shared key assignment. IEEE International Conference on Network-Based Information Systems (NBiS), pp. 263–270.

Ding, C., Pei, D. & Salomaa, A. 1996. Chinese remainder theorem: applications in computing, coding, cryptography. World Scientific.

Gutub, A., Al-Juaid, N. & Khan, E. 2017. Counting-Based Secret Sharing Technique for Multimedia Applications. Multimedia Tools and Applications, ISSN: 1380-7501, DOI: 10.1007/s11042-017-5293-6.

Gutub, A. 2006. Fast 160-Bits GF(p) Elliptic Curve Crypto Hardware of High-Radix Scalable Multipliers. International Arab Journal of Information Technology (IAJIT), Vol. 3, No. 4, pp. 342–349.

Iftene, S. 2006. Secret Sharing Schemes with Applications in Security Protocols", Sci. Ann. Cuza Univ, Vol. 16, pp. 63-96.

Kaya Kamer, 2009. Threshold Cryptography with Chinese Remainder Theorem", Diss. PhD thesis, Bilkent University, Department of Computer Engineering.

Morillo, P., Padro, C., Saez, G. & Villar J.L. 1999. Weighted threshold secret sharing schemes. Information Processing Letters, Vol. 70, No. 5, pp. 211-216.

Mignotte, M. 1982. How to share a secret. In Cryptography- Proceedings of the Workshop on Cryptography, Springer, Vol.149 pp. 371-375.

Rukhin, A. et al, 2001. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Booz-Allen and Hamilton Inc., McLean VA.

Sorin & Iftene, 2007. General secret sharing based on the chinese remainder theorem with applications in e-voting. Electronic Notes in Theoretical Computer Science, Vol. 186, No. 2, pp. 67–84.

Shamir, A. 1979. How to share a secret. Communications of the ACM, Vol. 22, No. 11, pp. 612- 613.

Simmons, G.S. 1992. An introduction to shared secret and/or shared control schemes and their application. Contemporary cryptology.

Tassa, T. 2007. Hierarchical threshold secret sharing. Journal of Cryptology, Springer, Vol. 20, No. 2, pp. 237–264.