Journal of Computer Science & Computational Mathematics, Volume 8, Issue 4, December 2018
DOI: 10.20967/jcscm.2018.04.006

Reliable Secret Key Generation For Counting-Based Secret Sharing

Adel Al-Qurashi*, Adnan Gutub

Computer Engineering Department, Umm Al-Qura University, Makkah, Saudi Arabia
*Corresponding author email: adel.qurashi8@gmail.com

***Abstract*:** Secret sharing scheme is becoming famous for increasing the security to access sensitive information for applications and resources that have to be protected by more than one person. It divides the secret key into shares, distributed to participants such that only subsets of participants can reconstruct the secret key. This paper adopts counting-based secret sharing scheme as a promising secret sharing technique presented recently. This counting-based method works on constructing shares by changing one or two 0-bits within the secret key to one at different locations for producing needed shares. The combination of selected shares is counting the ones in parallel to recover the secret key. This paper proposed improving the reliability of this counting-based secret sharing scheme by increasing the size of the secret key to 64-bits as realistically needed by most applications. The work also improved the security of the secret key affecting its shares by adjusting the generation algorithm to involve applicable statistical randomness tests from NIST 800-22 standard where any unreliable secret key is regenerated again whenever needed. The proposal is implemented and compared with the original scheme via Java platform modeling showing interesting practical remarks providing remarkable contributions.

***Keywords*:** information security, secret key generation, secret sharing, frequency Monobit test, frequency test within a block.

1. Introduction

Technological advancement in information technologies, communication networks, databases, and multimedia have become the nerve of knowledge life, industrial, financials, health, and security. Given to the increasing demand for web applications and the continuous growth of a number of networks users involving Cloud computing, E-Government, E-Commerce, E-Business, online banking services, which all contain sensitive information, it may be vulnerable to theft or access by wrong persons or even destroyed by hackers or service providers as non-trusted agencies in many cases. Therefore, there is an urgent need for information security with its three main components: Confidentiality, Integrity, and Availability (CIA) [1].

Information security is a big challenge in recent years, during which security has become the significant aspect of protecting from all the threats. There are several techniques of information security which have been proposed to protect data from disclosure. Some of them focus on traditional encryption to data, which is based on mathematical concepts and special arithmetic operations [2], while others focus on enhancing the confidentiality of the exchanged data by hiding the information in any cover media such as picture, sound, and text, known as steganography [3]. In fact, the security techniques, i.e. cryptography and steganography, focus on the phenomena that one person is in the controlling seat who is in charge of the secrecy of information [1]. However, the question is what happens when the encrypted or hidden data are corrupted, or the secret key is lost. It means that there is only security, but there is not reliability. Secret sharing considers solving this problem, as it allows to achieve high levels of confidentiality and reliability arbitrarily [4]. Secret sharing opened the computing filed for several interesting applications and resources which requires the security to be implemented by several participants, i.e., the security decision is shared among a set of participants.

In the beginning, the primary motivation behind the secret sharing was safeguarding cryptographic key from loss [5]. Losing a cryptographic key is an equal to data loss, as we cannot recover data without the encryption key. Keeping the cryptographic key at one location deos not depend on any physical or electronic problem, such as loss of key, system breakdown, sabotage or sudden death of persons who owns the secret key, etc. Losing the key leads to loss of data and may make access to the data impossible. Therefore, the storing of several key copies, i.e. distributing the cryptographic key in various sites, was the solution to increase the reliability of security access to systems. However, this distribution leads to decreasing confidentiality may be by making the status worst, causing main data to be in great risk to be lost, modified, destroyed, or leaking to wrong hands. Therefore, secret sharing considers solving this problem by achieving high level of confidentiality as well as reliability. Another motivation behind the secret sharing is the reduction of trust on a single person. The dominance of a specific authority avoids individual trusty by keeping the secret key by one person to control the secrecy of information alone [4]. Therefore, the secret sharing provides collective access to the confidential information by making the decision collectively and distributing the trust among many participants to further enhance the reliability and confidentiality.

To stress on the idea, the secret sharing is considered more needed in the areas that have highly sensitive information with big impact on decision making [6]. Accordingly, secret sharing can be seen via many examples in the real-life. It can be essential for opening the vault in central banks, nuclear missile launch control, voting systems, sensitive encryption keys, and considerable medical agreement. Each of these

cases requires access to be collectively agreed upon the sensitive information and resources. So, there is a need for this secret key to be shared or distributed among a set of participants by asking them all or a subset to be available at the same time for proper access [1].

Secret sharing scheme divides secret among a set of participants where that specified set of participants can reconstruct the secret. Therefore, the secret sharing scheme consists of two phases: the construction-distribution phase, and the secret-reconstruction phase. In the construction-distribution phase, the secret sharing scheme allows a trusted dealer (Algorithm) to generate secret shares by making information related to the secret called shares, and then these shares are distributed among a set of the participants via secure channel where each participant holds one share [7]. In the secret-reconstruction phase, the access structure enables a qualified subset of participants to collaborate and reconstruct the secret by collecting their shares in a specific way [8]. Note that secret sharing scheme divides all participants into two sets: the set able to retrieve the secret called authorized set and another set unable to recover the secret called an unauthorized set [9].

The secret key is considered very secure in secret sharing scheme. The system distributes the shares among users, n participants, where k or more of the participants ($k \leq n$) can retrieve the secret key by combining their k shares. The authorized set k is a subset of n participants (k out of n), considering (n, k) as threshold of secret-sharing scheme. In the threshold scheme, an attacker cannot discover the secret key, except if he knows at least k shares; knowing less than k shares should not reveal any information about the secret key [8]. In the secret sharing scheme there are two main properties which must be provided for applicability of any secret sharing scheme known as recoverability and confidentiality as declared below.

- Recoverability: It is the ability to retrieve the secret by the authorized set of participants by combining their secret shares.
- Confidentiality: It is the fact that no shares can lead to disclosure of information of the secret key, i.e. stopping any intruder attempt to recover the secret key by guessing from less than k shares.

Threshold secret sharing scheme began in 1979 by Shamir [6] and Blakley [5] independently. Shamir's scheme depends on polynomial interpolation, while Blakley's scheme depends on the geometry. Since then many work attempts have been proposed to serve secret sharing schemes. Some of them handled problems improving Shamir [6] and Blakley [5] methods and some others innovated new techniques, all to achieve the same principle secret sharing scheme, i.e. by various methods, such as Kai Wang [10]; Tassa [11]; Herzberg [12]; Komargodski [13]; Bai [14]; Blundo [15]; Chi-Sung Laih, Tzonelih Hwang [16], and lately Adnan Gutub et al. [1]. This research focuses on Adnan Gutub's novel approach of counting-based secret sharing working on constructing the shares by changing one or two 0-bits at different specific locations within the secret key for producing a new share. This counting-based secret sharing scheme have been studied due to its wide range of applicability to almost all applications as well as its simplicity and performance in its running. Hence, the combination of shares can be applied by parallel counting the ones within selected shares (k shares) to recover the secret key.

This counting-based secret sharing scheme simplicity and practicality come from the method low mathematical operations usage, i.e. when constructing shares until retrieving the secret key, as well as its achievement of reasonable level of security, as will be stressed upon later in this work. Accordingly, in this paper, we have proposed improving the reliability of the counting-based secret sharing scheme by increasing the size of the secret key to 64-bit and applying two statistical tests on the generated secret key checking randomness of the shares and secret key sequence. This strategy allows getting the optimal reliability of secret key and constructing its shares contributing to the improvement of the security system.

The main objectives of this reliable secret key generation study can be stated in three points.

- increasing the reliable secret key size to 64-bits to male it suitable for real-life applications.
- testing the security of the reliable secret key as randomly generated via passing NIST RNG testing.
- verifying whether the reliable secret key is able to serve the number of users being capable to produce enough trustworthy shares.

The paper has been organized as follows. Section 2 covers the related works about secret sharing schemes. Section 3 presents specific background about the counting-based secret sharing scheme. We cover generating the shares and retrieving the secret key via clarification examples. Section 4 presents our proposal modeling the reliable secret key selection within counting-based secret sharing. Section 5 discusses comparisons and result analysis. The final section, Section 6, includes the conclusion and the recommendations of this study.

2. Related Work

In the literature, there are many researches who discussed Threshold Secret Sharing Schemes from several aspects. For example, Shamir's Threshold Secret Sharing Scheme is the first scheme for secret sharing proposed by Adi Shamir in 1979 [6]. It is based on Lagrange interpolation polynomial. To get (t, n) threshold scheme, D pick a random (t-1) degree polynomial $q(\times) = a_0 + a_1\times + \ldots\ldots + a_{t-1}\times^{t-1}$ in which a_0 is the secret $K \in \mathbb{F}_P$ and $a_i \in \mathbb{F}_P$, where p is a prime number. Dealer generates n shares $Si = q(i)\ldots\ldots\ldots$ $S_n = q(n)$ and then distribute them to the participants by channel secure. By selecting any subset t of those S_n values, we can construct the polynomial by Lagrange interpolation and recover the secret, but cannot calculate the secret with (t-1) participants. Shamir's threshold scheme has some features that makes it secure. The size of shares does not override the size of the secret key. Also, the scheme is to be expandable, i.e. when t is fixed, the shares can be dynamically deleted or added without impact on the other shares. The system can have

some changes within the shares without alteration of the secret key as well as without needing to generatie new shares to the participants. Shamir's scheme is thought to be a hierarchical scheme, by which it can provide each participant with a various number of shares based on their importance inside the organization [6].

In Blakley's scheme [5], the system is dependent on geometry to achieve a (t, n) threshold, the secret is represented as a point P in the vector space $\mathbb{F}_q^t$, and n shares are distinct (t-1) dimensional hyperplanes that pass through this point p that contain the secret, where (t−1)-dimensional hyperplane is a set of form:{ $(\mathcal{X}_1,\ldots,\mathcal{X}_t) \in \mathbb{F}_q^t \ |\alpha_1\mathcal{X}_1 + \ldots\alpha_t\mathcal{X}_t = \beta$ } where $\alpha_1,\ldots,\alpha_t$ and β are arbitrary points of the field $\mathbb{F}_q$. The secret can be obtained by intersecting t hyperplanes at P, whereas fewer than t hyperplanes will intersect only in some subspace containing P. Thus fewer than t participants are able to recover the subspace, but cannot recover the secret correctly. In fact, Blakley's scheme is not preferred due to the fact that unauthorized group of participants may know partial information of the subspace containing the secret making them able to guess the secret. The scheme is then improved by Simmons [17] to make it acceptable using affine space instead of vector space.

However, in multi-level organizations, there is the need to share secret among all the members of the organization in hierarchically structured groups, where members from different levels have varying powers and as such, they need to factor in the powers in sharing the information [18]. Sharing of information is based on a predetermined sequence of threshold requirements. Such thresholds require the presence of a member with a higher power to enhance the organization's secret, as in Tamir-Tassa's Hierarchical Secret Sharing in [11]. Tamir-Tassa's introduced the perfect secret sharing scheme to solve users' problem in their threshold secret sharing scheme, where the secret is shared among a group of users into their levels. This proposal uses polynomial derivatives to generate lesser shares for participants of lower levels, compared with Shamir's scheme in which the secret is represented as the free coefficient of some polynomial. Thus, the secret is reconstructed in this scheme by using Birkhoff interpolation as discussed to assign identities of the participants from the underlying finite field [11].

Based on Herzberg et al. [12], the proactive secret sharing scheme is a game changer in secret sharing. Traditionally secret sharing schemes relied on fixed shares. However, in long-lived and sensitive secrets, this approach is not sufficient, as attackers may gain access to enough shares to reach the set threshold before the life of the secret is over. A proactive scheme of secret sharing solves this situation by periodically renewing shares. This is done without changing the secret being protected. If an attacker has accessed to any portion of the shares using the old information, the information becomes useless as soon as an update is done. In this scheme, an attacker has limited time to break into k locations before an update occurs [12].

Bai et al. [14] improved Shamir's single-secret sharing scheme to a multiple-secret sharing scheme using matrix projection. They proposed a proactive secret sharing scheme method to renew (n) secret shares periodically in a (k, n) threshold-based secret sharing scheme without changing the secret or reconstructing the secret to generating new shares. Also, he presented a distributed proactive secret sharing scheme for the matrix projection secret sharing scheme. Note that, his technique cannot reveal the secrets from (k) shares by adversaries when new mixed shares with past and present are updated (i.e., this method is protected against the passive attacks) [14].

In Asmuth-Bloom Secret Sharing Scheme [19], the scheme uses an ascending sequence for a set of pair wise co-prime positive integers ($x_0 < x_1 < x_2 < .. < x_n$, where $x_0 > K$ is a prime), are chosen such that: $x_0 \cdot \prod_{i=0}^{K-2} x_{n-i} < \prod_{i=1}^{K} x_i$. Asumuth-Bloom scheme works on chooses secret K as a random integer from $\mathbb{Z}x_0$, and then n shares are constructed as $S_i = (K + r * x_0) \bmod x_i$, for all $(1 \le i \le n)$ where r is a positive integer generated randomly such that $K + r * x_0 \in \mathbb{Z}x_0 \cdot x_k$. Given k various shares $S_1; S_2, \ldots, S_K$, the secret K is retrieved as $K = S_0 \bmod x_0$, , where x0 is the unique solution of the system of congruencies using Chinese remainder theorem CRT [19].

A Fully Dynamic Secret Sharing Schemes was presented by Blundo et al [15]. They presented a study with different access structures and proposed that a dealer enables a particular set of users to reconstruct different secrets by sending the same broadcast message to all users. This approach is based fully on information-theoretic without any computational assumption, i.e. the security of the scheme is unconditional. Also, the model appears both the size of shares held by users and the size of the broadcast message based on the defined size of the secret.

Evolving Secret Sharing Dynamic Thresholds and Robustness presented by Komargodski et al. [13] depended on proposing an efficient scheme for secret sharing among an unlimited number of participants, where only subsets of (k) participants can recover the secret. They evolved their method to resolve the problem of an efficient scheme for the dynamic threshold access structure. The method considered the size of qualified groups increasing as the number of participants increases which showed how to translate any scheme for k-threshold into the scheme which is robust. This secret sharing system had security incompatibility such that its secret can be recovered even if some participants have incorrect shares [13].

3. Overview on Counting-Based Secret Sharing Scheme

The counting-based secret sharing scheme works by generating all the possible shares from the secret key (SK) [1]. It is mainly using two methods, namely 1-bit and 2-bit methods, which both work in different styles. The generated shares can be denoted by A having the same size as the SK secret key. This scheme only chooses n shares as being useful, i.e., n out of A shares are found to be suitable, while the remaining shares should be ignored. It is important to consider these n shares from the beginning to correctly distribute them among participants in secure way by

authentic channel or trusted dealer. Note that these applicable n of A shares are selected accurately to be combined with each other to reconstruct SK fulfilling the counting-base reconstruction strategy. However, in case one or more of n shares are absent or unavailable, SK cannot be regenerated. Thus, subset of k shares is to be assigned of n, where k≤ n is able to reconstruct SK. This counting-based secret sharing scheme can be classified as (n, k) threshold scheme. It is to be mentioned that the security of the system relies tremendously on the difficulty of reconstructing SK from shares found to be less than k [1].

This section will introduce the original two methods of shares generation, 1-bit and 2-bit methods, which are affecting the pool A consisting acceptable and unacceptable shares. Then, the secret key SK retrieval from secret shares will be briefly presented clarifying the counting-based secret sharing approach. A clarification example is provided to elaborate the idea covering different cases of real applicable scenarios.

3.1 Generating Shares via 1-Bit Method

Secret shares generating via the 1-bit method depend mainly on the zero bits within SK. This 1-bit method generates all possible A shares based on the number of zeros found within SK, i.e. shares cannot be more than the number of zeros. This method works on selecting one zero from a specific position of SK and then flipping it to one to produce a valid share. Note that every time in generating the new shares, we must select the different location not previously chosen for producing another share and so forth.

The following example clarifies the 1-bit method to generate shares, where we proposed a simple example of SK= [1 0 0 1 0 0 1 0] as presented in Table 1. Note that in this example, SK has 5 zeros, so it generates five shares by changing one zero at a time, as in highlighted cells in Table 1. This method is considered simple to implement, fast to run, confirmed as reliable, and all shares are useful when performing the combination in parallel, but it has a drawback where it gives a limited number of shares produced.

Table 1. Example of the 1-Bit method shares construction

SK	1	0	0	1	0	0	1	0	92 Hex
Sh1	1	1	0	1	0	0	1	0	D2
Sh2	1	0	1	1	0	0	1	0	B2
Sh3	1	0	0	1	1	0	1	0	9A
Sh4	1	0	0	1	0	1	1	0	96
Sh5	1	0	0	1	0	0	1	1	93

3.2 Generating Shares via 1-Bit Method

The 2-bit method is a modification to increase the numbers of shares generation described by means of the 1-bit method. The 2-bit method can be used alone or as an extension with the 1-bit method to increase the numbers of shares. This 2-bit method depends on changing two zeros within SK to generate extra possible shares. The method scans SK for zeros, whenever two zeros are found, they can be flipped providing a probable share to be used. Not all generated 2-bit shares can be useful since some that are generated do not fulfilling the counting-based secret key SK reconstruction strategy.

For our example of SK= [1 0 0 1 0 0 1 0] , we can flip the first zero with second zero to generate a new share as shown in Table 2. Similarly, we can flipp the first zero with third zero to generate more new shares and so on. These applicable 2-bits shares are used in addition to the 1-bits shares for generating all the possible shares A within SK.

Table 2 illustrates a modified example of Table 1 showing extra shares of 6 to15 added over the 1-bit method as an extension. The number of shares generated by this 2-bit method is 10, i.e. by flipping two different zeros every time within different positions. Therefore, the total shares generated by combining the two methods for SK= [1 0 0 1 0 0 1 0] are 15 shares, A=15 shares. Note that the 2-bit method enhances the 1-bit method and increases the number of shares produced for SK. However, not all these A shares are useful. It is found that many shares are not suitable to reconstruct SK, this factor forces us to verify the validity of needs A to generate n suitable shares. So, n out of A shares are to pass the testing to reconstruct SK before using them to ensure the applicability before distributing among the participants.

Table 2. Example of the 2-Bit & 1-Bit methods shares construction

1 – Bit Method									
SK	1	0	0	1	0	0	1	0	92 Hex
Sh1	1	1	0	1	0	0	1	0	D2
Sh2	1	0	1	1	0	0	1	0	B2
Sh3	1	0	0	1	1	0	1	0	9A
Sh4	1	0	0	1	0	1	1	0	96
Sh5	1	0	0	1	0	0	1	1	93
2 – Bit Method									
Sh6	1	1	1	1	0	0	1	0	F2
Sh7	1	1	0	1	1	0	1	0	DA
Sh8	1	1	0	1	0	1	1	0	D6
Sh9	1	1	0	1	0	0	1	1	D3
Sh10	1	0	1	1	1	0	1	0	BA
Sh11	1	0	1	1	0	1	1	0	B6
Sh12	1	0	1	1	0	0	1	1	B3
Sh13	1	0	0	1	1	1	1	0	9E
Sh14	1	0	0	1	1	0	1	1	9B
Sh15	1	0	0	1	0	1	1	1	97

3.3 Secret Key Retrieval from Secret Shares

As discussed before, the counting-based secret sharing scheme cannot use all A shares to recover SK. Only selected n shares are to be reliable and acceptable for distribution to participants. Therefore, n shares need to be tested to make sure of their validity before their usage as set of authorized participants shares, i.e. good to retrieve SK. Nevertheless, in case that one or more of participant's shares are found absent, SK cannot be recovered. Consequently, a subset of n which is used as prerequisite to reconstruct SK must be provided as k shares. This subset k is less than or equal to n ($k \leq n$) and preserving the same condition of it cannot recover SK if the shares are less than k inputted. After determining n shares and making sure of their validity, the application defines a value of k out of n to be used as threshold of available true users' shares in parallel for SK regeneration. Accordingly, the k shares are mapped in parallel within the system and the

parallel bits are counted. If the counting output from shares parallel combination for all bits in any column equals the value of k or more, then the resulted bit is one otherwise the resulted bit is zero, and so on. These resulted bits are combined and compared with original SK, i.e. to check the validity of the shares for SK secret key reconstruction. The reader is referred to a study [1] for more in-depth elaboration on the philosophy behind this counting-based secret sharing scheme.

3.2.1 Clarification Examples

The following examples focus on the counting based secret sharing idea and its SK secret key proper generation. The examples illustrate different cases for utilization of shares to clarify SK retrieval method and possible challenges to be addressed. The example is made simple with the same 8-bits SK size used in Table 1 for clarification purpose. The research will show a real-life application using 64-bits key as studied later. Recall SK= [1 0 0 1 0 0 1 0] as introduced in Table 1 and the shares generated by 1-bit and 2-bit methods exist in Table 2 assuming the number of selected shares to be given to users as n=8, as follows:

	A	Binary SK								Hex SK
1-bit Method	Sh1	1	1	0	1	0	0	1	0	D2
	Sh2	1	0	1	1	0	0	1	0	B2
	Sh3	1	0	0	1	1	0	1	0	9A
	Sh4	1	0	0	1	0	1	1	0	96
	Sh5	1	0	0	1	0	0	1	1	93
	Sh6	1	1	1	1	0	0	1	0	F2
2-bit Method	Sh7	1	1	0	1	1	0	1	0	DA
	Sh8	1	1	0	1	0	1	1	0	D6
	Sh9	1	1	0	1	0	0	1	1	D3
	Sh10	1	0	1	1	1	0	1	0	BA
	Sh11	1	0	1	1	0	1	1	0	B6
	Sh12	1	0	1	1	0	0	1	1	B3
	Sh13	1	0	0	1	1	1	1	0	9E
	Sh14	1	0	0	1	1	0	1	1	9B
	Sh15	1	0	0	1	0	1	1	1	97

Choosing n_{Sh} → n = 8

Sh1	1	1	0	1	0	0	1	0	D2
Sh2	1	0	1	1	0	0	1	0	B2
Sh4	1	0	0	1	0	1	1	0	96
Sh5	1	0	0	1	0	0	1	1	93
Sh7	1	1	0	1	1	0	1	0	DA
Sh8	1	1	0	1	0	1	1	0	D6
Sh9	1	1	0	1	0	0	1	1	D3
Sh15	1	0	0	1	0	1	1	1	97

Case 1: Situation of Combining Shares = k

In this case, it is assumed that k=4 and the number of shares = k. This case is considered valid. Thus, the shares are combined as in Table 3.

Table 3. The Example of the Situation of Combining Shares = k

Sh1	1	1	0	1	0	0	1	0	D2
Sh2	1	0	1	1	0	0	1	0	B2
Sh7	1	1	0	1	1	0	1	0	DA
Sh8	1	1	0	1	0	1	1	0	D6
Counting results	4	3	1	4	1	1	4	0	k=4
SK	1	0	0	1	0	0	1	0	92

Consider the counting result row which counts the ones within every column. SK reconstruction is performed by assigning value one whenever counting column result ≥k, otherwise a zero is placed in that location. Thus, the retrieved outcome of the combination process equals hexadecimal value 92 which is correct as SK=92.

Case 2: Situation of Combining Shares > k.

In this case, it is assumed that k=4, and the number of shares is greater than k. Thus, this case is considered valid even if the number of shares is more than k. The shares are combined with the condition that the counting result of bits is greater than or equal to k, as in Table 4.

Table 4. The Example of the Situation of Combining Shares > k

Sh1	1	1	0	1	0	0	1	0	D2
Sh2	1	0	1	1	0	0	1	0	B2
Sh5	1	0	0	1	0	0	1	1	93
Sh8	1	1	0	1	0	1	1	0	D6
Sh9	1	1	0	1	0	0	1	1	D3
Sh15	1	0	0	1	0	1	1	1	97
Counting results	6	3	1	6	0	2	6	3	k=4
SK	1	0	0	1	0	0	1	0	92

Note, in this case, the number of shares is more than k. So, if the counting result of bits in one column ≥ k, then it gives one, otherwise, zero is placed in that location. Thus, the hexadecimal outcome of the combination process is 92, i.e. as needed SK=92.

Case 3: Situation of Combining Shares < k.

In this case, assuming k=4, and a number of shares is less than k, thus it is considered unable to retrieve SK. As needed, it cannot recover the SK secret key as the number of shares be less than k, as in Table 5.

Table 5. The Example of the Situation of Combining Shares < k

Sh1	1	1	0	1	0	0	1	0	D2
Sh5	1	0	0	1	0	0	1	1	93
Sh7	1	1	0	1	1	0	1	0	DA
Counting results	3	2	0	3	1	0	3	1	k=4
SK	0	0	0	0	0	0	0	0	00

Note that in this case, the number of shares is less than k. Thus, the outcome of the combination process is 00 ≠ SK=92, which is exactly as required.

Case 4: Situation of Involving Intruder False Share

Assume an intruder inserts a false share in the same scenario of Case 1, i.e. k=4, and a number of shares = k, but one of the shares (or more) is falsely inserted by an intruder, as in Table 6.

Table 6. The Example of the Situation of Involving one Intruder as False Share

Sh1	1	1	0	1	0	0	1	0	D2
Sh2	1	0	1	1	0	0	1	0	B2
Sh7	1	1	0	1	1	0	1	0	DA
FSh	1	0	0	0	1	0	0	1	89
Counting results	4	2	1	3	2	0	3	1	k=4
SK	1	0	0	0	0	0	0	0	80

This case is depicted to be invalid, where all the shares are valid except the false one marked as FSh of hexadecimal value 89. Thus, the outcome of the combination process is 80 ≠ SK=92, confirming the system security validity.

Assume the same case of intruder inserting false shares but more than one, namely FSh1 and FSh2, as in Table 7.

Table 7. The Example of the Situation of Involving Intruder Multi False Shares

Sh1	1	1	0	1	0	0	1	0	D2
Sh2	1	0	1	1	0	0	1	0	B2
FSh1	1	0	1	1	1	1	0	1	DB
FSh2	0	1	1	0	1	1	0	0	6C
Counting results	3	2	3	3	2	2	2	1	k=4
SK	0	0	0	0	0	0	0	0	00

These two false shares involvement result in invalid SK as needed. Thus the outcome of the combination process is found to be 00 ≠ SK=92 that verifies the scheme security strength.

4. The Proposed Reliable Secret Key Selection

This paper is proposed to improve the original counting-based secret sharing scheme though increasing the reliability of the secret key selection process. The aim is to increase the shares space applying the method on secret key of size SK 64-bits compared to the original scheme of small SK as for clarifying the idea of counting-based secret sharing. The original method depends on the simple variant sizes of SK, i.e. SK=4-bits, SK=5-bits, SK=6-bits, SK=8bits, and SK=12-bits. These sizes are not realistic to be used in real-life applications. So, we proposed to study the SK binary format of 64-bits allowing the scheme to be adopted in reality password numbers, such as: SK= [10110010 00110101 01101101 00110001 00110110 00110011 01100110 01110010] and are represented in Hex as SK= [B2 35 6D 31 36 33 66 72].

Recall the principle phenomena of SK secret key in counting-based secret sharing scheme to be unknown to all participants. Only the system dealer (Algorithm) is to know the SK secret key. This means that the SK secret key sequence should be random and preferred impossible to be guessed or predicted. This difficulty of guessing the random secret key sequence by intruders is the main block assuring secrecy of the system. So, the system dealer (Algorithm) generates SK secret key with its size 64-bit using Random Number Generators (RNGs), i.e. to produce pseudo-random SK secret key. Therefore, we assume an RNG is used at the beginning to provide SK followed by security verification before producing the shares, as algorithm flow graph shown in Figure 1. The SK secret key generated by RNG needs to be verified to be realistic. This SK sequence should be random and possible to provide number of shares enough for the users, which is a process needed before selecting the shares by the 1-bit and 2-bits methods. To check the randomness of SK in this proposed scheme, we will rely on applying two statistical standard tests from NIST 800-22 suites to test the randomness of SK. The aim is getting a reliable random SK secret key. This proposed method introduced the new reliable counting-based secret sharing scheme, as depicted in Figure 1.

In this new approach, we tested the modified algorithm using an Intel processor Core i7 PC with speed 2.90 GHz, RAM 16 GB, 64-bit operating system. Also, we depended on NetBeans IDE platform version 8.9 as our programming environment for simulating of the counting-based secret sharing scheme via Java language platform. This platform is used for simulating the two RNG tests of NIST 800-22 which are Frequency (Monobit) test and Frequency test within a block to provide reliable SK results. In addition, we depended on database of MySQL Workbench version 6.3 as our storage memory to keep the results and we link them with NetBeans IDE. Extracting the results have been performed by accessing the database MySQL Workbench and then exporting data to Excel program for analysis and comparison.

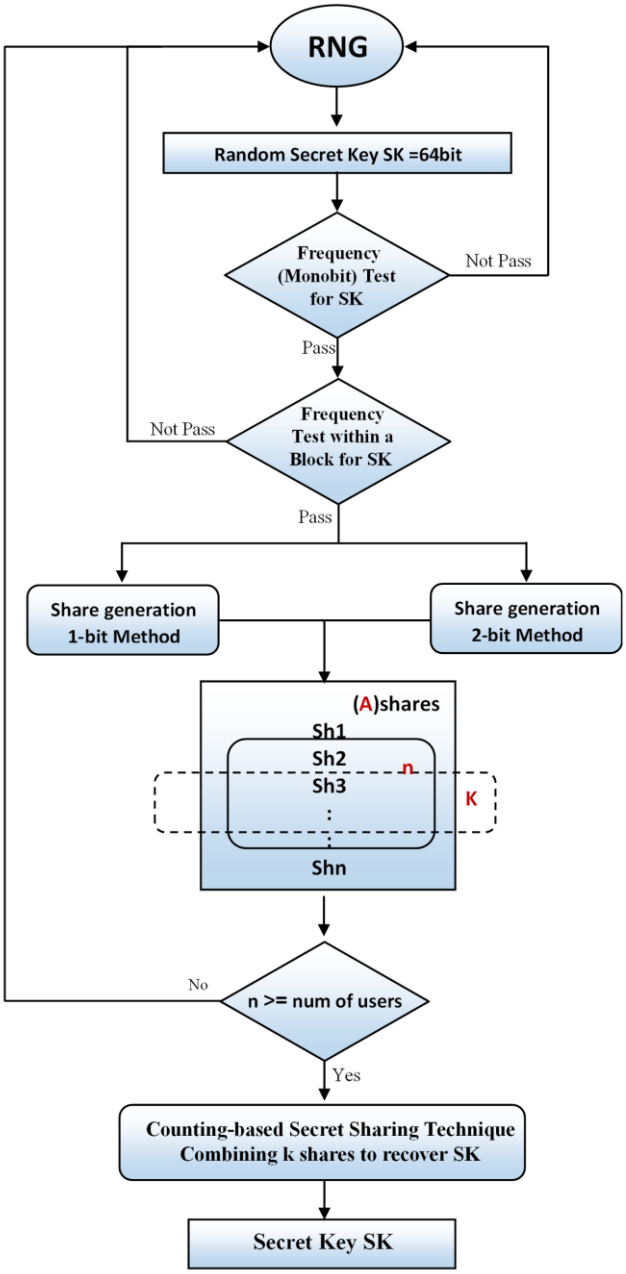

Figure 1. Proposed Modified Counting-Based Secret Sharing

4.1 Generating Reliable Random Secret Key

Random numbers are basically known as sequence of numbers that are almost unpredictable in nature. It is assumed to be generated through Random number generator (RNG) which provide random sequence of numbers having no particular order or pattern to form them [20]. In the literature, there are two basic types of random number generators used to produce random sequences, True Random Number Generator (TRNG), Pseudo-Random Number Generator (PRNG) [21].

4.1.1 True Random Number Generator (TRNG)

TRNG is an electronic piece that plugs into a computer and produces random numbers from a physical process, rather than computer programs [21]. TRNG uses a non-deterministic source (i.e., the entropy source) to produce randomness and often depends on measuring the unpredictable process of microscopic phenomena that generate random noise signals, such as thermal noise, atmospheric noise, the photoelectric effect, the quantum effects in a semiconductor, etc. The outputs of these random processes are assumed completely unpredictable, it may be used directly as a random number or may be supplied as a seed into a pseudo-random generator (PRNG). In fact, the generation of high-quality random numbers is considered unrealistic, too much time-consuming, making TRNG undesirable when a large quantity of random numbers is needed. So the research always recommends to use pseudorandom number generators to generate large number of random numbers [22].

4.1.2 Pseudo-Random Number Generator (PRNG)

PRNG is an algorithm used to produce a sequence of numbers which it is not truly random, but its properties are closer to the properties of sequences of random numbers [22]. PRNG depends on an initial value, called a seed to generate multiple pseudo-random numbers. This seed should be random and unpredictable. So the pseudo-random numbers of a PRNG are deterministic, i.e., all true randomness is confined to seed generation. The pseudo-random numbers are periodicity which is a desirable feature for several applications, like simulations of stochastic processes, statistical sampling and performance assessment of computer algorithms and Monte Carlo simulation. PRNG is important in several fields for its speed in random number generation compared to TRNGs which are comparatively slow [23].

In this proposed scheme, we depended on PRNG function from Java program to generate the pseudo-random number for SK. This Java program provides support to generate random numbers primarily through (java.util.Random classes) [24]. The random function will generate a set of random bits based on the required 64-bits range intended as reliable size of SK. Therefore, these pseudo-random numbers of secret keys that are produced by the random function are subject to statistical tests verification which aim to emphasize the randomness of the secret key sequence. Reliable SK is then used to construct shares via the 1-bit and 2-bit methods, i.e. based on the number of zeros existed as shown earlier in Fig 1.

4.2 Statistical RNG Tests

Statistical tests provide a mechanism for comparing and evaluating the sequence of bits as making standard decisions to determine the sequence randomness. The tests aim to try to verify that the random sequence of bits does not follow a definite pattern or specific order and it cannot be described as a probabilistic property [21]. There are a large number of probable statistical tests, each of them is evaluating the existence or non-existence of a pattern by a certain way. If RNG testing outcomes is found acceptable, then it will indicate that the sequence is applicably random to be accepted [25]. Statistical tests are formulated to test a specific null hypothesis (H0). In this paper, the null hypothesis indicates that SK sequence is being tested as

random. If this case is not achieved, the hypothesis requests an alternative hypothesis (Ha) as set, which indicates that SK sequence is non-random. When any statistical test is applied, a decision or conclusion is obtained that accepts or rejects the null hypothesis, i.e. whether the generated SK is considered acceptably random or not. The possible outcome of the statistical hypothesis testing, either accept H0 assuming the secret key is random, or reject H0 asking for the secret key to be regenerated again. This rejected decision is called Type I error.

The possibility of Type I error is usually called the level of significance of the test denoted as α. That means α is the probability that the test will reference the SK sequence as non-random when it indeed is random. The reason may be that SK sequence holds non-random properties even when being the generator as pretended reliably good.

The statistic test is used to compute a P-value that determines the force of the evidence against the null hypothesis. Hence, P-value is the probability that RNG would have generated SK sequence less random than the sequence tested. If a P-value for a test converges to 1, then SK sequence seems to be perfect close to applicable randomness. Otherwise, if P-value test indicates value near zero, then SK sequence seems to be completely non-random. To be realistic, the system needs to decide an acceptable range for P-value to consider SK randomness acceptable, which is decided by significance level α. This significance level α is our realistic threshold that can determine the realistic tests within the range [0.01 or 0.001]. If (P-value ≥ α), we consider the null hypothesis as accepted, i.e., SK sequence assumed to be random. If (P-value < α), we conclude that the null hypothesis is rejected and α indicates the probability of Type I error, i.e., SK sequence seems to be non-random. We decided to accept 1% as the level of significance α, so the test specified in this study needs 0.01 minimum to be accepted. This made the reality acceptance rate to be of 1 SK from 100 SK, which insured reliable security. The system is tuned to only allow for a P-value ≥ 0.01, pretending the secret key would be considered random (reliable) with a confidence of 99.9%. Accordingly, we chose two applicable NIST 800-22 standard tests, namely the frequency (Monobit) test and the frequency test within a block, to determine whether SK is reliable in terms of randomness [21], as introduced next. In fact, the reason to choose these two tests is to emphasize considering all the bits (frequency Monobit test) as well as portions of blocks (frequency test within a block) having reliable number of ones and zeros. For this case, the number of zeros within SK shouldn't be much more than the number of ones, where it can be observed clearly through previous elaboration shares example of Table 2. On the other hand, as the number of ones increases within SK, the number of shares generated reduces and makes a different contradicting variable, which may lead to unacceptable reliability identifying SK as invalid to be used, too.

Our work selected the applicable two tests for verifying randomness of SK which have been implemented within the proposed system providing reliable remarks.

4.2.1 Frequency (Monobit) Test

The Frequency (Monobit) test focuses on testing the proportion of ones and zeros for the entire sequence of the SK secret key. This test works on determining whether the numbers of ones and zeros in a sequence are nearly equal. In fact, all tests within NIST 800-22 standard depend on the passing of this Monobit test, i.e., the success of this test gives reliability evidence for the existence of randomness in the SK secret key sequence and allows for possible success of the other test. If this test fails, the generated SK is rejected, as shown in Figure 1.

4.2.2 Frequency Test within a Block

Frequency test within a block is sub-derived from the previous Monobit test to further stress on security reliability. This test asks for diving the sequence (SK) into M-bit blocks. The focus then, becomes on the proportion of ones within this M-bit blocks. It works on determining whether the frequency of ones in any block is approximately M/2, nominating the sequence of the secret key in accepted reliability randomness. It is assumed as for the reason to choose Frequency Test within a Block to give homogenous distribution closer to equal among ones and zeros within SK to insure its fairness. This test checks for cases where any part of SK is containing more number of ones or zeros than the others. Therefore, failing this test makes the probability of guessing SK from generated shares high and then leads to low reliability.

These two tests are ready tools libraries provided by NIST. We applied them within our modified secret sharing procedure aiming to calculate the P-value, i.e. to determine randomness and then to enable us to acquire the reliable secret key. To summarize, if P-value ≥ 0.01 then SK indicates that the sequence is reliably random and the SK secret key is valid to use for constructing shares. Otherwise, the secret key is not-secure to use and the algorithm (Figure 1) asks for new SK to be generated.

5. Comparison and Analysis

The proposed reliable counting-based secret sharing system is implemented via Java platform. This model results are analyzed at each stage to verify its contribution. We show remarks of the statistical tests applied to SK to acquire reliability within the secret key in order to proceed further within the process (Figure 1) to construct the shares. We studied the proportion of ones and zeros in SK based on the statistical tests to determine preferred SK secret key to be used. Therefore, our analytical study has been based on random samples of SK, which are generated by Random Function within our Java program generating series of pseudo-random bits, as observed in Table 8. These pseudo-random samples of SK have been subjected to the two statistical tests from NIST 800-22 suites, i.e. the frequency (Monobit) test and the frequency test within a block, in order to experiment their randomness. We simulated these two tests applied on the samples to get the P-value per SK which determines whether the sequence for SK is random or non-

random, then sorted them as presented in Table 8. To simplify readability, Table 8 random SKs have been listed from SK_1 to SK_{63} as providing reliable randomness, while the non-random samples of SKs, generated by the same Java Random Function, have been sorted as SK_{64} to SK_{100} to prove possible non-random results.

Table 8. Randomness Tests for Generated Secret Keys

ID	Secret Key for Hex	Num of Zeros	Frequency Test		Frequency Test within a Block	
			P-value	Result	P-value	Result
k1	11F24EDB5DF13604	32	1	Rand	0.25807	Rand
Sk 2	E05E7C3FCDA5279C	28	0.317311	Rand	0.84799	Rand
Sk 3	8EFACEEB0BDE0163	29	0.453255	Rand	0.169963	Rand
Sk 4	C6C9D03A54DBD908	34	0.617075	Rand	0.423763	Rand
Sk 5	40514DFCB317682A	35	0.453255	Rand	0.377154	Rand
Sk 6	179BF475C4D6F891	29	0.453255	Rand	0.891292	Rand
Sk 7	4BEA6A76C0F2D525	31	0.802587	Rand	0.799347	Rand
Sk 8	B1E3258E374F319E	30	0.617075	Rand	0.927926	Rand
Sk 9	1499C854EF674C59	33	0.802587	Rand	0.377154	Rand
Sk10	6A09BB374A03CF49	33	0.802587	Rand	0.29423	Rand
Sk11	15CBB4CAC20FD5E4	32	1	Rand	0.981012	Rand
Sk12	A782CDCF42CE9ABD	29	0.453255	Rand	0.29423	Rand
Sk13	9C15E6DA1482F44F	33	0.802587	Rand	0.580338	Rand
Sk14	9C2FEF5B68A5A89E	28	0.317311	Rand	0.525883	Rand
Sk15	64367EA57FC03EBD	27	0.2113	Rand	0.169963	Rand
Sk16	77DDC689112BDF0E	29	0.453255	Rand	0.169963	Rand
Sk17	D5C1B2C4261A9601	38	0.133614	Rand	0.525883	Rand
Sk18	FA17B82397CC052F	31	0.802587	Rand	0.691937	Rand
Sk19	DE613179549D2FB2	30	0.617075	Rand	0.746837	Rand
Sk20	CC02DD9120213D41	40	0.0455	Rand	0.040971	Rand
Sk21	BE28FA3B385DB570	29	0.453255	Rand	0.377154	Rand
Sk22	8C91AD930A2E4110	40	0.0455	Rand	0.258077	Rand
Sk23	2F5FDE8C2DBFF627	23	0.024449	Rand	0.169963	Rand
Sk24	ED509BC4962D794A	32	1	Rand	0.636031	Rand
Sk25	75F95277FC2ADE0D	26	0.133614	Rand	0.258077	Rand
Sk26	91DA68DE69A09CDE	31	0.802587	Rand	0.473485	Rand
Sk27	2D68073619A89E36	35	0.453255	Rand	0.956905	Rand
Sk28	12AB8D9B6799EE97	28	0.317311	Rand	0.636031	Rand
Sk29	EF30BC41756093AD	32	1	Rand	0.146798	Rand
Sk30	D21A3312F010CF2E	36	0.317311	Rand	0.33393	Rand
Sk31	C48733170FC20E2E	35	0.453255	Rand	0.992708	Rand
Sk32	8640BF06FB1A507E	33	0.802587	Rand	0.008289	Rand
Sk33	FD32F2DA9E368C43	29	0.453255	Rand	0.473485	Rand
Sk34	1CB8189CE132DCA8	36	0.317311	Rand	0.84799	Rand
Sk35	76BB881970120FEA	34	0.617075	Rand	0.423763	Rand
Sk36	C96FCAC7237B8F51	28	0.317311	Rand	0.636031	Rand
Sk37	91B36CE1950CECC0	35	0.453255	Rand	0.691937	Rand
Sk38	D407141615A0C1DC	39	0.080118	Rand	0.580338	Rand
Sk39	854B78391A5F4DA3	32	1	Rand	0.934358	Rand
Sk40	3B6C9DA2EBF3BA21	28	0.317311	Rand	0.423763	Rand
Sk41	FFCA398066C2572D	31	0.802587	Rand	0.095765	Rand
Sk42	C201036AD7494BFB	34	0.617075	Rand	0.079196	Rand
Sk43	FA7191C146461C4E	35	0.453255	Rand	0.799347	Rand
Sk44	D8ADC09623105CE6	36	0.317311	Rand	0.423763	Rand
Sk45	A2DA40516B7E00F4	36	0.317311	Rand	0.029084	Rand
Sk46	FCA881B7ED86B8D9	29	0.453255	Rand	0.29423	Rand
Sk47	C69C682CA1ECB562	34	0.617075	Rand	0.927926	Rand
Sk48	6CF10AEF913CCECB	29	0.453255	Rand	0.377154	Rand

Sk49	962D32681FDD2968	33	0.802587	Rand	0.799347	Rand
Sk50	9A292DF2C52B71E9	31	0.802587	Rand	0.992708	Rand
Sk51	ED8D3C2B25C3A749	31	0.802587	Rand	0.891292	Rand
Sk52	D2617CE1CF8C88E2	33	0.802587	Rand	0.691937	Rand
Sk53	E49864EC2AD4EBB3	31	0.802587	Rand	0.799347	Rand
Sk54	58CBAA2C311F2762	34	0.617075	Rand	0.927926	Rand
Sk55	7393A2892CEE41D1	34	0.617075	Rand	0.636031	Rand
Sk56	3B4E38EE12A63A5C	32	1	Rand	0.746837	Rand
Sk57	11E69633E9CC53B0	33	0.802587	Rand	0.891292	Rand
Sk58	214AB18A73EEAAF0	33	0.802587	Rand	0.691937	Rand
Sk59	3D0B7B463472783F	30	0.617075	Rand	0.636031	Rand
Sk60	1C87B2247D964FB6	31	0.802587	Rand	0.691937	Rand
Sk61	70D6872AD85DC6A5	32	1	Rand	0.981012	Rand
Sk62	559AA8174F5E7E92	30	0.617075	Rand	0.84799	Rand
Sk63	B24FD832373B1887	32	1	Rand	0.84799	Rand
Sk64	FDEFD7BA9F9A6EFF	16	0.000063	Non-Rand	0.004734	Non-Rand
Sk65	FFEDDFA7B7FFAFF3	12	0.000001	Non-Rand	0.000305	Non-Rand
Sk66	CDBDF8DA7BFAF7FD	17	0.000177	Non-Rand	0.034554	Rand
Sk67	AEBFF65DEE1FECFD	18	0.000465	Non-Rand	0.05723	Rand
Sk68	FFFF8E7BBF3CDD6F	15	0.000021	Non-Rand	0.000829	Non-Rand
Sk69	24032040A016808C	49	0.000021	Non-Rand	0.008289	Non-Rand
Sk70	352600002840EBA2	45	0.001154	Non-Rand	0.001229	Non-Rand
Sk71	D080004051083316	48	0.000063	Non-Rand	0.003238	Non-Rand
Sk72	6C18208801521202	48	0.000063	Non-Rand	0.009964	Non-Rand
Sk73	0000530500C02710	51	0.000002	Non-Rand	0.000076	Non-Rand
Sk74	0819005080400321	52	0.000001	Non-Rand	0.000456	Non-Rand
Sk75	0000181811082B01	52	0.000001	Non-Rand	0.000135	Non-Rand
Sk76	A0080A19B611000B	46	0.000465	Non-Rand	0.009964	Non-Rand
Sk77	528800B138D07000	46	0.000465	Non-Rand	0.009964	Non-Rand
Sk78	00210C00C0481209	52	0.000001	Non-Rand	0.000456	Non-Rand
Sk79	4512A40B00300005	49	0.000021	Non-Rand	0.002674	Non-Rand
Sk80	0080600040000401	58	0	Non-Rand	0.000001	Non-Rand
Sk81	1400268832043A01	48	0.000063	Non-Rand	0.004734	Non-Rand
Sk82	8110528A4410AA14	46	0.000465	Non-Rand	0.040971	Rand
Sk83	0880092E44100030	51	0.000002	Non-Rand	0.000557	Non-Rand
Sk84	0001007081021108	54	0	Non-Rand	0.000039	Non-Rand
Sk85	0080090062000081	56	0	Non-Rand	0.000002	Non-Rand
Sk86	FFF7B6A42BBF3DFF	17	0.000177	Non-Rand	0.000829	Non-Rand
Sk87	3F771FB7FEFCEF95	17	0.000177	Non-Rand	0.024434	Rand
Sk88	FFAF6EE515FDFFFD	15	0.000021	Non-Rand	0.000373	Non-Rand
Sk89	7FDDA4FFAAD7DFFF	15	0.000021	Non-Rand	0.000249	Non-Rand
Sk90	F6EE7FD5FAEFB7FE	14	0.000007	Non-Rand	0.004734	Non-Rand
Sk91	D66BBCF7F926FF77	19	0.001154	Non-Rand	0.017152	Rand
Sk92	96DC3E55FDDEDBBD	21	0.00596	Non-Rand	0.169963	Rand
Sk93	BEC2EDBDFDBF73F2	19	0.001154	Non-Rand	0.034554	Rand
Sk94	11ACFFAF7FAEFF1E	20	0.0027	Non-Rand	0.001495	Non-Rand
Sk95	4D97711FDBFF7CFB	20	0.0027	Non-Rand	0.040971	Rand
Sk96	FF3316FF24F90CB9	26	0.133614	Rand	0.003238	Non-Rand
Sk97	C47D2E76EFDBE7FF	19	0.001154	Non-Rand	0.01196	Rand
Sk98	7B9DEBB7DFFC5ACB	19	0.001154	Non-Rand	0.092806	Rand
Sk99	66279B37FA7BEDFF	20	0.0027	Non-Rand	0.05723	Rand
Sk100	7108E8604000E92A	44	0.0027	Non-Rand	0.009964	Non-Rand

5.1 Selecting Reliable SK Based on Frequency Test

Frequency (Monobit) test focuses on the proportion of ones and zeros for the entire SK sequence. The numbers of ones

and zeros in a sequence are to be verified closer to equal. So, this test results in Table 8 compute the absolute value of the sum of ones and zeros S_n within SK sequence. Where, ones represented for +1 and zeros for -1, and then the resulting absolute value of the sum of ones and zeros in SK is divided by the square root of the size of SK sequence of 64 bits. Then, we will get on the test statistic value $S_{obs}= \frac{S_n}{\sqrt{n}}$. As mentioned earlier, the level of significance α determined in Frequency (Monobit) test is 0.01. Accordingly, the computation of P-value through erfc (denoting to the complementary error function), gives P-value = erfc ($\frac{S_{obs}}{\sqrt{2}}$). Accepting H0 if the P-value ≥ α then SK would be considered reliably random. This test has been applied to the 100 sample of SK, where p-value has been calculated per SK based on the proportion of ones and zeros in the entire SK sequence. When the number of ones and zeros is close to equal, they will cancel each other. Thus that test statistic result will be almost 0 and the P-value will be equal to one. That means that this SK is perfectly random based on this test and valid to use as the reliable secret key in the counting-based secret sharing scheme.

In fact, the counting-based secret sharing scheme depends on the number of zeros within the secret key to construct secret shares. Since the proposed SK size in counting-based secret sharing scheme is 64 bit, then the equal point in this test between ones and zeros within SK sequence is 32, and the P-value indicates to 1. It means that the SK which contains 32 zeros, will be perfectly random and reliable. The population of testing P-values of SKs can be shown in Figure 2 below.

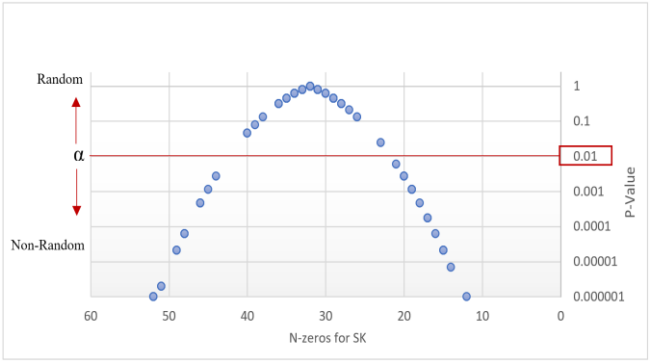

Figure 2. P-value Distribution of Randomness Applying Monobit Frequency Test

Consider Figure 3, observing the random samples of secret keys passed the frequency test, i.e. having p-value > 0.01, we will determine the reliable random secret key as valid to use in counting-based secret sharing scheme.

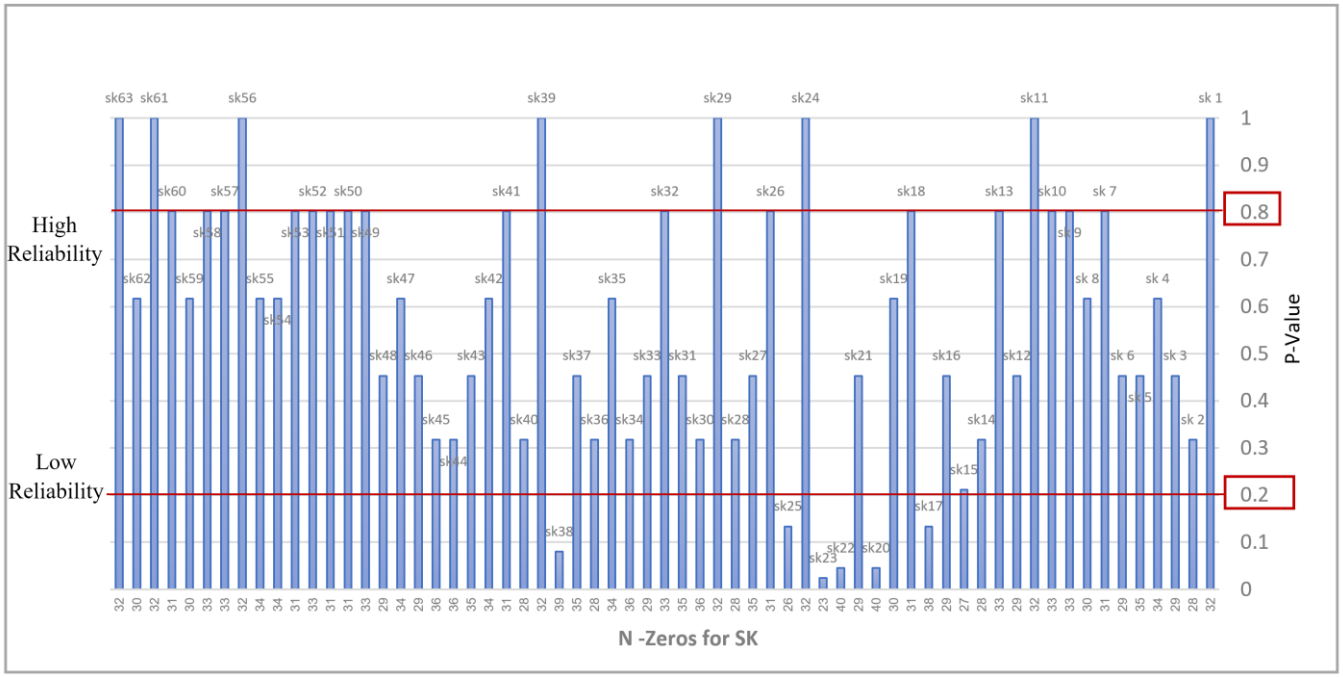

Figure 3. Secret Keys Reliability Based on Frequency Test

In Figure 3, we have assumed that if P-value ≥ 0.8, i.e. when the number of zeros within SK equals 31, 32 & 33, then the random secret key has a high level of reliability to be used, as the case in (sk_1, sk_7, sk_9, sk_{10}, sk_{11}, sk_{13}, sk_{18}, sk_{24}, sk_{26}, sk_{29}, sk_{32}, sk_{39}, sk_{41}, sk_{49}, sk_{50}, sk_{51}, sk_{52}, sk_{53}, sk_{56}, sk_{57}, sk_{58}, sk_{60}, sk_{61}& sk_{63}). Similarly, if the P-value < 0.2, as the number of zeros within SK equals 26, 25, 24, 23, 40,39 & 38, then the secret keys hold a low level of reliability and are not advisable to be used, as the case in (sk_{17}, sk_{20}, sk_{22}, sk_{23}, sk_{25}& sk_{38}). As for, the remaining of secret keys in which P-values are range between 0.8 and 0.2 they hold a medium level of reliability and can be used.

5.2 Selecting Reliable SK Based on Frequency Test within a Block

Frequency test within block focuses on the proportion of ones within M-bit blocks, and through determining whether the frequency of ones in an M-bit block approaching of M/2 until as would be the secret key is accepted as randomness. This test works on partitioning SK sequence into N= $[\frac{n}{m}]$

non-overlapping blocks, where n indicates the length of SK and m the number of bits within a block. Any unused bits are discarded. In this frequency test within a block, the null hypothesis H0 defines that SK sequence is random. The alternative hypothesis Ha is SK sequence that is non-random. Test statistic χ computes the proportion π_i of ones in each block, i.e., the number of ones within the block is divided by m. The test statistic χ is represented in the equation: $\chi^2 = 4\,M\sum_{i=1}^{N}(\pi_i - \frac{1}{2})$.

Recall the level of significance α as determined before in this study to be as 0.01. Accordingly, we can compute the P-value through IGAMC (denoting to Incomplete Gamma Function), where P-value = IGAMC (N/2, χ^2/2). Accordingly, accepting H0 if the P-value ≥ α then SK would be considered random. Reject H0 if the P-value < α then SK would be considered non-random. The testing P-values of SKs can be shown in Figure 4 below.

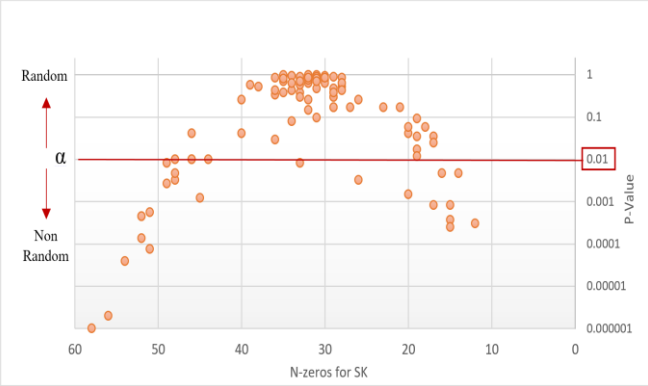

Figure 4. P-value Distribution of Randomness Applying the Frequency Test Within Block

Figure 4 results generated are from testing the 100 samples of SKs, where SK length is 64-bit (n=64), and the number of bits within a block is 8-bit (M=8). Observe that all the points that represent acceptable SKs, i.e. from SK_1 to SK_{63} (Table 8), are located above the level of significance α (i.e., all p-values ≥ 0.01). Hence, these secret keys considered are random. That means accepting the null hypothesis H0 is that SK is random. Through Figure 4, there are many SKs that have not overridden the frequency test within a block, as listed in Table 8 which are (sk_{64}, sk_{65}, sk_{68}, sk_{69}, sk_{70}, sk_{71}, sk_{72}, sk_{73}, sk_{74}, sk_{75}, sk_{76}, sk_{77}, sk_{78}, sk_{79}, sk_{80}, sk_{81}, sk_{83}, sk_{84}, sk_{85}, sk_{86}, sk_{88}, sk_{89}, sk_{90}, sk_{94}, sk_{96}, sk_{100}). That means that these SK secret keys are not-random (i.e., all p-values < 0.01). Consequently, the null hypothesis H0 is rejected and Ha hypothesis is accepted alternative as that the SK is non-random. Note that, some of SKs have overridden the test as in Table 8, i.e. (sk_{66}, sk_{67}, sk_{82}, sk_{87}, sk_{91}, sk_{92}, sk_{93}, sk_{95}, sk_{97}, sk_{98}& sk_{99}). These SKs appear to have acceptable randomness while not passing the Monobit test, due to the reason of holding some of random properties when divided into small blocks.

Figure 5 represents random samples of SKs which passed the frequency test within a block having p-value ≥ 0.01. These samples determine the reliable random SK secret key valid to use in counting-based secret sharing scheme. The test represents great importance to get on distributing the bits to be ones and zeros within SK as closer to equal percentages.

Figure 5 also shows SKs for different higher P-value ≥0.8. The secret key has a high level of reliability as common case in (sk_2, sk_6, sk_8, sk_{11}, sk_{27}, sk_{31}, sk_{34}, sk_{39}, sk_{47}, sk_{50}, sk_{51}, sk_{54}, sk_{57}, sk_{61}, sk_{62}& sk_{62}). Controversially, if the P-value ≤ 0.2, for secret key holding low level of reliability is not advisable to use, as the case in (sk_3, sk_{15}, sk_{16}, sk_{20}, sk_{23}, sk_{29}, sk_{32}, sk_{41}, sk_{42} & sk_{45}). Acceptable usage can be for the remaining P-values range between 0.8 and 0.2 holding medium level of reliability.

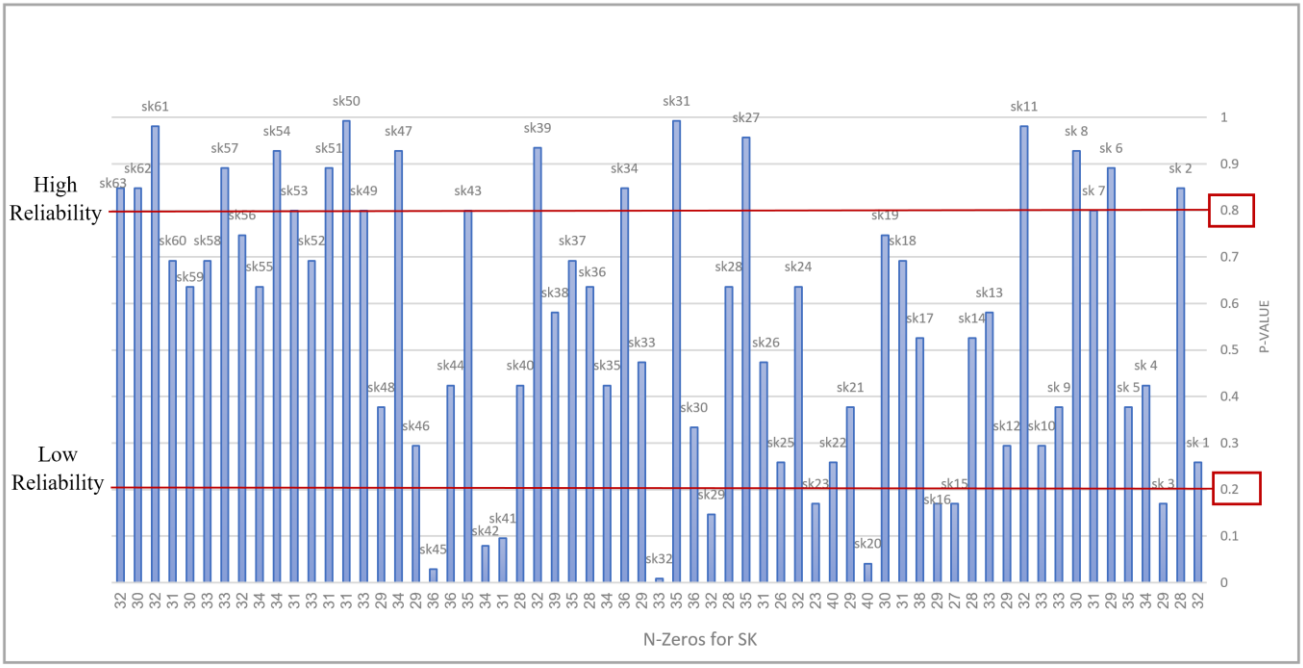

Figure 5. Random Secret Keys Reliability Based on Frequency Test within Block

5.3 Practical Reliable Secret Key Selection

The randomness frequency tests alone are not enough for SK practicality. The tests are limited to compute the proportion of ones and zeros within SK sequence which tends to converge. Therefore, having SK with passing statistical test values, i.e., P-value ≈1, can lack the optimal distribution of ones and zeros within the sequence, i.e. degrading the optimal reliability. It can be observed as part of SK containing more number of ones or zeros than other parts. Accordingly, higher P-value is recommended to avoid this problem. We found that any SK passed both tests and having P-value for both tests greater than or equal to 0.8, i.e. P-value ≥ 0.8, considers the selection practically optimal for the reliable secret key, which is preferred to be used in the counting-based secret sharing scheme. Combining both tests within one graph can give proper indication. Consider Figure 6, the F-test denotes Monobit frequency test and the F-Test within Block denotes the other frequency test within a block showing the SKs passing with high P-values, as the case in (sk_{11}, sk_{39}, sk_{50}, sk_{51}, sk_{57}, sk_{61}, sk_{63},), where their P-value ≥ 0.8. These SKs above 0.8 can be considered preferred selection for the reliable secret key although they are found only 7% of the SK generated.

Figure 6 show that some of SKs have achieved their P-value ≥ 0.8 in one test, but less in the other one and vice versa. Therefore, any SK passing 0.8 only with one test is not considered preferred optimal selection although considered reliable, as the case in (sk_1, sk_2, sk_6, sk_7, sk_8, sk_9, sk_{10}, sk_{13}, sk_{18}, sk_{24}, sk_{26}, sk_{27}, sk_{29}, sk_{31}, sk_{32}, sk_{34}, sk_{41}, sk_{47}, sk_{52}, sk_{53}, sk_{54}, sk_{56}, sk_{58}, sk_{60}, sk_{62}), representing 25% of the generated SKs. The low level of α = 0.2 or 0.1 (i.e. 0.1 < p-value < 0.2) is achieving low reliability of SK and preferably should not to be used although its percentage is 31%, which is homogenous to the non-reliable percentage of 37% non-passing randomness making a challenge to generate preferred SK secret key sequences.

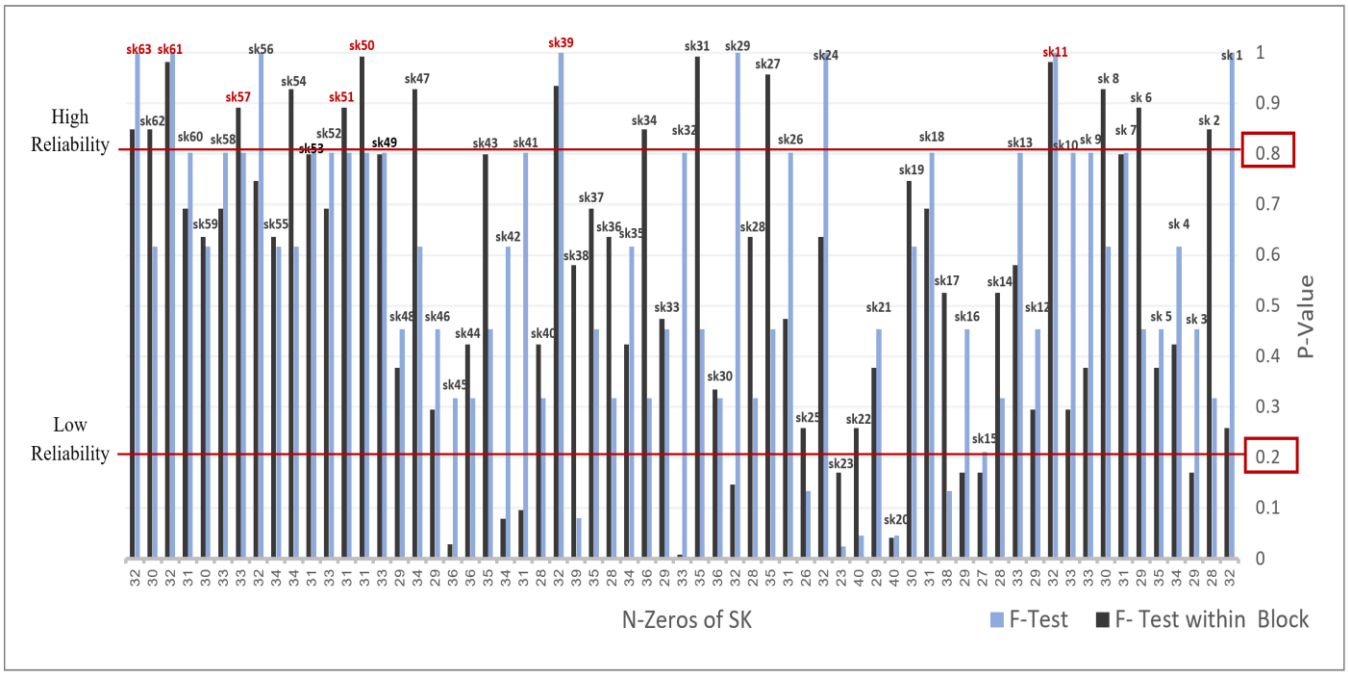

Figure 6. Practical Optimal Selection for Random Secret Keys Reliability

6. Conclusion

In this work, we have increased reliability to improve the counting-based secret sharing scheme by increasing the size of the SK to 64-bits as being practical for real life applications. We improved the secret sharing algorithm by making the SK sequence generated and verified for randomness. It is believed that this secret key SK generation made intruders guessing difficult to succeed. The study focused on checking SK randomness to be reliable and further practically optimal. Therefore, the test depended on Java Random function generator to produce the pseudo-random sequence testing for 100 SK samples.

The randomness has been experimented applying two standard statistical tests from NIST 800-22 suites, Frequency (Monobit) test and Frequency test within a Block, showing interesting randomness features. However, it is found that the best random SK valid to use is when the number of zeros and ones within it are almost equal, i.e. approaching 32 for our study 64-bit SK size. Yet, the number of zeros tests are not enough alone to fully make the system reliable for secret key in the counting-based secret sharing scheme. It is possible to find SK containing 32-zeros but lack the optimal distribution of ones and zeros within the sequence, hence SK is commonly containing more number of ones or zeros in one part than another. This challenge justified applying the frequency test within a block further to the Monobit frequency test which helped getting more reliable random secret keys.

The reliability study is assumed to be the highest when SK achieves both frequency tests together with the significance level α = 0.8 (i.e., p-value ≥ 0.8); thus, we conclude that SK

has high reliability and consider optimized the best to be used in the counting based secret sharing scheme to construct secret shares. The analysis and comparisons also considered different randomly generated SK samples showing medium and low reliability level interesting outcomes. The research determined two more levels of significance combing both reliability randomness tests showing high, medium, and low preference. The high reliability level sets the focus on significance $\alpha = 0.8$ (i.e. p-value ≥ 0.8) which achieved limited 7% SKs, recommended as fully reliable to be used. The low level of $\alpha = 0.2$ or less (i.e. $0.1 <$ p-value < 0.2) is achieving low reliability of SK and preferred not to be used although it's found in large numbers of SKs with percentage of 31%. The medium level of P-values ranges between 0.8 and 0.2 is holding reasonable reliability and can be considered as it is found 25% of the SK space.

Future work

We suggest recommendations to improve this paper in the future by increasing the size of SK to 128-bit. Also, we recommend applying other statistical tests from NIST 800-22 test suites on the random secret key, for getting on the optimal, reliable secret key to use in counting-based secret sharing scheme.

Acknowledgment

I would like to thank my wonderful supervisor Prof. Adnan Gutub who supported me a lot in accomplishing this humble work successfully according to his guidance. I consider myself very lucky, because. I have learned and benefited a lot from him in the field of scientific research. Also I would like to thank Umm Al-Qura University for providing me this golden chance to continue my higher study to get the Master's Degree in the field of computer studies.

References

[1] A. Gutub, N. Al-Juaid, E. Khan, "Counting-Based Secret Sharing Technique for Multimedia Applications", *Multimedia Tools and Applications*, Springer, DOI 10.1007/s11042-017-5293-6, pp. 1-29, 2017.

[2] A. Gutub, A. Tenca, "Efficient Scalable VLSI Architecture for Montgomery Inversion in GF(p)", *Integration the VLSI Journal*, vol. 37, no. 2, pp. 103-120, 2004.

[3] E. Ahmadoh, A. Gutub**,** "Utilization of Two Diacritics for Arabic Text Steganography to Enhance Performance", *Lecture Notes on Information Theory*, vol. 3, no. 1, pp. 42-47, 2015.

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

[5] G. R. Blakley, "Safeguarding cryptographic keys", *Proc. of AFIPS National Computer Conference*, vol. 48, pp. 313-317, 1979.

[6] A. Shamir, "How to share a secret", *Communications of the ACM 22*, vol. 22, no. 11, pp. 612- 613, 1979.

[7] K. Alaseri , A. Gutub, "Merging Secret Sharing within Arabic Text Steganography for Practical Retrieval", *IJRDO - Journal of Computer Science and Engineering*, vol. 4, no. 9, 2018.

[8] S. Iftene, "Secret Sharing Schemes with Applications in Security Protocols", *Sci. Ann. Cuza Univ*, vol. 16, pp. 63-96, 2006.

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

[10] K. Wang, X. Zou,Y. Sui, "A Multiple Secret Sharing Scheme based on Matrix Projection", *Proc. of the 33rd Annual IEEE International Computer Software and Applications Conference*, pp. 400-405, 2009.

[11] T. Tassa, "Hierarchical threshold secret sharing", *Journal of Cryptology*, vol. 20, no. 2, pp. 237–264, 2007.

[12] A. Herzberg, S. Jarecki, K. Hugo, M. Yung, "Proactive Secret Sharing Or: How to Cope With Perpetual Leakage", *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '95)*, London, UK: Springer-Verlag, pp. 339–352, 1995.

[13] I. Komargodski, A. P. Cherniavsky, "Evolving Secret Sharing: Dynamic Thresholds and Robustness", *Theory of Cryptography Conference*, Springer, Cham, pp.379 – 393, 2017.

[14] L. Bai, X. Zou, "A proactive secret sharing scheme in matrix projection method", *International Journal of Security and Networks*, vol. 4, no. 4, pp. 201-209, 2009.

[15] C. Blundo, A. Cresti, A. De Santis, U. Vaccaro, "Fully dynamic secret sharing schemes", *Theoretical Computer Science*, vol. 165, pp. 407-440, 1996.

[16] C. S. Laih, L. Harn, J. Y. Lee, T. Hwang, "Dynamic Threshold Scheme Based on the Definition of Cross-Product in an N-Dimensional Linear Space", *Proceedings of Advances in Cryptology (CRYPTO)*, Lecture Notes in Computer Science, vol. 435, pp. 286-298, 1990.

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

[18] A. Castiglione, A. De Santis, B. Masucci, "Hierarchical and shared key assignment", *17th IEEE International Conference on Network-Based Information Systems (NBiS)*, pp. 263-270, 2014.

[19] C. Asmuth, J. Bloom, "A modular approach to key safeguarding", *IEEE transactions on information theory*, vol. 29, no. 2, pp. 208-210, 1983.

[20] K. Ikake, "Random number generator", U.S. Patent, no. 7,124,157, 2006

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

[22] L. Liang, *Testing Several Types of Random Number Generator*, The Florida State University, 2012.

[23] W. Janke, "Pseudo random numbers: Generation and quality checks", *Lecture Notes John von Neumann Institute for Computing*, vol. 10, p. 447, 2002.

[24] Random (Java Platform SE 8), Java Platform Standard Edition 8 Documentation. Available: https://docs.oracle.com/javase/8/docs/api/java/util/Random.html.

[25] R. Smaliukas, *Block Cipher and Non-Linear Shift Register Based Random Number Generator Quality Analysis*, Vilnius University Institute of Mathematics and Informatics, 2015.