Multimed Tools Appl (2019) 78:5591–5619
https://doi.org/10.1007/s11042-017-5293-6

Counting-based secret sharing technique for multimedia applications

Adnan Gutub[1] · **Nouf Al-Juaid**[2] · **Esam Khan**[3]

Received: 9 July 2017 / Revised: 1 October 2017 / Accepted: 5 October 2017 /
Published online: 2 November 2017

Abstract Secret Sharing is required in situations where access to important resources has to be protected by more than one person. We propose new secret-sharing scheme that works based on parallel counting of the ones within the shares to generate the secret output. Our work presented two different modeling variations that are mainly different in the secret-sharing keys generation where both are studied elaborating their pros and cons. Our counting-based secret shares key reconstruction is implemented and simulated considering the security level required by the usage functions. Comparisons showed interesting results that are attractive to be considered. This secret sharing method is of great benefit to all multimedia secret sharing applications such as securing bank sensitive accounts and error tracking, voting systems trust, medical agreements, wills and inheritance authentication management.

Keywords Secret sharing · Key management · Shares generation · Information security · Key distribution technique

1 Introduction

Due to the increase demand on information technology and multimedia communication, information security is being very important aspect. Many techniques have been

✉ Adnan Gutub
aagutub@uqu.edu.sa

Nouf Al-Juaid
naljuaid@su.edu.sa

Esam Khan
eakhan@uqu.edu.sa

[1] Computer Engineering Department, Umm Al-Qura University, Makkah, Saudi Arabia

[2] Shaqra University, Riyadh, Saudi Arabia

[3] The Custodian of Two Holy Mosques Institute for Hajj and Umrah Research, Umm Al-Qura University, Makkah, Saudi Arabia

developed and proposed to increase the security of information [18]. Some focused on scrambling the multimedia information not to be useful such as cryptography based mainly on mathematical concepts and special arithmetic operations [12] based on different Galois Fields (GF) arithmetic [6]. Other security techniques focused on hiding the information itself such as steganography, which also showed different variations based on the hiding schemes [1]. Some implemented the security features on general software platforms while others dedicated hardware modules for it, showing interesting security achievements [7]. These security techniques are focusing on the idea that one person is controlling the secrecy of information. Nowadays, some applications are found popular requiring security to be performed by several persons, where technical proof of sharing the secrecy is a must. This idea is known as the Secret-sharing [19], which is one of these techniques that are getting more attention among security, cryptography, and steganography area. It is needed for many applications where essential software or e-resources has to be really secured or cannot be reveled easily except by physical agreement and human interaction [3].

Secret sharing schemes are becoming more important for storing information that is highly sensitive with big affect [21]. Examples include encryption keys, missile launch control, and numbered bank accounts, where access of each of these must be kept collectively top confidential, as their exposure could be disastrous. Traditional methods of encryption and normal cryptography achieve high levels of confidentiality and reliability [9], but lack the real collectivity access of decision making [19]. This is because when storing the encryption key, one alone must choose between keeping a single copy of the key in a location for maximum secrecy, or keeping multiple copies of the key in different locations for greater reliability [3]. In fact, increasing reliability of the key by storing multiple copies lowers confidentiality by creating additional attack possibilities. Also, there are more opportunities for a copy to fall into wrong participants hands. Secret sharing schemes address this problem specifically, and allow arbitrarily high levels of confidentiality and reliability to be achieved.

Secret sharing schemes are becoming important in cloud computing media. Thus shares of a key can be distributed over many servers by a threshold secret sharing mechanism. The key, i.e. target key, is then reconstructed as needed. In other words, secret sharing can be studied as the method of distributing the ownership of a secret target key amongst a group of participants, each of whom is allocated a share of this secret. The secret can be reconstructed only when a sufficient number of shares are combined together, whereas individual shares cannot be useful on their own [19].

The secret sharing scheme is a tool can be used in many cryptographic protocols. It is dedicated for assisting key management security and authentication [8]. In secret-sharing, a secret target-key *TK* is split into *n* useful shares, which are distributed by a dealer to a number of participants [21]. In theory, the target key *TK* can be reformed in A collection *A* of subsets, however, not all *A* is useful as appropriate secret shares for our *TK* combination function. The useful share from set *A* is labeled as *n*, where any threshold subset, i.e. *k* out of *n* can reconstruct the target-key TK from its shares. In order to reconstruct the target-key, these $k \in n$ shares must be merged together in a specific combination. The reconstruction main concern is that any group of *k* shares or more (*k* is the threshold) can together be merged to get the secret *TK,* but no group of fewer than *k* participants can. Also, repetition of a share is not allowed in the reconstruction process resulting in false *TK* output. Such a system is called a (*k, n*) -

or k out of n - secret sharing threshold scheme. The process of reconstructing the target-key *TK* from an access structure is called a combiner or *target-key TK reconstruction*. To be concise with the literature, the main two properties that any secret sharing scheme has to fulfill are:

- **Recoverability**: where the target-key *TK* can be reconstructed given any k shares.
- **Secrecy**: where no information can be known about *TK* given any number of shares $< k$.

The idea was first introduced by George Blakley [3] as well as Adi Shamir [19] both in 1979. Since then, Different efficient schemes were proposed as classified in the next section showing many different flavors. We propose a new secret-sharing scheme that works based on recovering the target-key *TK* via counting the ones of the k shares in parallel. The applicable k secret shares are placed with their bits in parallel allowing their ones to be counted, i.e. in parallel, making the resulting secret output one if the threshold is passed. The work details the method model and simulates it adopting two different secret shares generation techniques, i.e. focusing on 1-bit one, or 2-bits ones, where both are studied showing promising results.

The paper is organized as follows. Section 2 covers the related work literature survey. It discusses different classifications of secret sharing schemes followed by examples of secret-sharing applications presented in Section 3. The examples are given to show the great benefit of the proposed secret sharing technique to applications such as securing bank sensitive accounts and error tracking, voting systems trust, medical agreement, wills and inheritance authentication management. Section 4 proposes our counting-based secret-sharing scheme. Then, Section 5 presents the modeling and simulation that is elaborating on the two variations proposed for secret shares generation. Section 6 discusses the proposed method comparisons elaborating on the benefits and drawbacks of every model variation detailing their effect on the security level. Finally, the paper is concluded in Section 7.

2 Related background of secret-sharing schemes

Secret-sharing schemes can be classified based on five different bases, where some are related toward this research more than others. The five secret-sharing schemes can be classified based on number of shared secrets, based on share weight, based on the changeability of shares, based on the rights given to the dealer and participants, and based on the techniques used. Every classification scheme has its properties as briefly clarified in the following:

2.1 Secret-sharing based on number of shared secrets

This secret sharing classification based on number of shares is very basic and can be thought of as the ignition beginning this research area. The number of shares can reduce to only one or increase and require fair distribution to several participants. The single share is resulting into single target-key *TK* making it dealt with as normal simple symmetric key security system. On the other hand, more than one share, i.e. several secret shares method is the real beneficial approach motivating this entire secret sharing phenomenon, which is found motivating our proposed counting-based secret sharing technique.

To be precise, the literature shows that the secret-sharing can be classified based on number of shared secrets into:

- Single secret: where only the target-key *TK* can be shared.
- Multiple secrets: where the secret-sharing scheme allows multiple secrets to be shared.

Examples of single secret schemes include Shamir; Blakley; Mignotte; Asmuth-Bloom; Brickell; Ghodosi; Iftene; Benaloh; Feldman; Pedersen; Ingemarsson; Jackson; Martin; Steinfelda; Herzberg. Examples of multiple secret schemes include Chien; Yang; Shao; Franklin; Pang; He; Bai; and matrix projection [21].

2.2 Secret-sharing based on share weight

The secret sharing classification based on share weight giving some participants more power in target-key *TK* regeneration more than the others, which can known or can be kept confidential not to affect the others participation nor understanding. The analogy of this in real life can be understood in the voting system, for example, where the chairman is given the power (or weight) of two or three votes among the other participants. In other words, the secret-sharing can be classified based on share weight into:

- Same weighted shares: where all shares have the same weight.
- Multi-weighted shares: this can be done in two ways. The first is to assign more shares to some participants. Shamir's hierarchical secret sharing scheme is an example of this method. In the second way, some shares contain more information about the shared secret different than other shares. An example of such interesting approach is proposed by Tassa's scheme [20].

2.3 Secret-sharing based on the changeability of shares

Secret sharing classification based on changeability of shares affected by time combined to power of position as classified into three methods. First, for example, shares are made to expire and/or must be refreshed or regenerated after certain time, as is the common case nowadays with frequent password changing request in high security sensitive systems. This method is known here as proactive secret sharing. The second changeability secret sharing classification is the timely dynamically changing after distribution, of course adhering to certain agreement. The third classification is giving some discrimination among participants, i.e. some are given more power not to require changing their shares while others must change. This classification does not link to our proposed counting-based secret sharing technique since all participants are given the shares once not allowing any changing. To summarize this classification from out of the box, the three changeability secret-sharing methods can be briefed as follows:

- Proactive secret sharing: here, the shares are updated periodically. Old shares are not used any more. Herzberg scheme is an example of this class [13].
- Dynamic secret sharing: there exist two cases here. In the first, the dealer can change some access structures. In the second, some participants are given the ability to reconstruct different secrets. Two examples of this type are given in [5, 15].

- Secret sharing with veto capability: the secret key in this class is prevented from being reconstructed. The research in [4] gives an example of this type.

2.4 Secret-sharing based on the rights given to the dealer and participants

This secret sharing classification is based on the rights given to the dealer and participants allowing reconstruction of shares based on honesty and trust. Some, i.e. the dealer and/or participants, can be given the arithmetic foundation tool to regenerate the shares as required. As this level of trust reduces, the computational tool details are kept confidential among them forbidding the recovery possibility which is out of the scope of our proposed counting-based secret sharing technique. For the completion of the study, the classification of secret-sharing based on the rights given to the dealer and participants can be briefed as follows:

- Computational secret sharing: the computation power of the participants (and maybe the dealer as well) is bounded. For example, in the Krawczyk technique [14], the information rate allowed for a participant equals the ratio between the average length of the share and the length of the secret.
- Verifiable secret sharing: in this technique, it is supposed that not all participants are honest. Therefore, honest participants should be able to recover the secret and dishonest ones should not get any information on it. Ogata et al. [17] adopted Shamir's scheme to be able to detect dishonesty in secret sharing.
- Robust secret sharing: this technique allows reconstruction of the secret in the case of corrupting some shares due to some adversary actions. An example of such techniques is given in [16].

2.5 Secret-sharing based on the techniques used

The secret-sharing can be classified depending on the arithmetic techniques used. It can also involve hardware vs. software features such as its implementation on pipelined e-security dedicated modules [8], or efficient crypto-arithmetic adders [11], avoiding the software based on polynomial or interpolation computations as well as the known Chinese Remainder Theorem [2]. It can also be computed anonymously as will be the case followed in our research proposing this counting-based secret sharing technique.

To be concise, the secret-sharing can be classified based on techniques used into:

- Polynomial based secret sharing: which is based on polynomials and interpolations, such as Lagrange's interpolation [19] and Brikhoff interpolation [20].
- Chinese Remainder Theorem (CRT) based secret sharing: For example, in [2], a specific number of participants can reconstruct the secret by using CRT. In fact, CRT arithmetic can benefit from dedicated hardware with software features such as the expandable modulo multiplier presented in [10].
- Anonymous secret sharing: in this type, the secret can be reconstructed without knowing which participants holds which share.

3 Applications of secret-sharing

There are many applications that can benefit from secret sharing. This section will list some of them especially applicable to our proposed counting-based secret sharing technique, as the following different real life examples.

3.1 Bank sensitivity accounts and error tracking

Consider sensitive bank accounts that need at least k shares from a total of n shares to reconstruct the super-password, i.e. target-key TK, to deal with important customers or companies' accounts. It can also be used to correct human mistakes or common errors in the banking system in entering any information, which is needed to increase the accuracy percentage and show record for the error causing assistants. This system is also needed whenever a large financial deduction is to occur making the banking top management question what is happening.

3.2 Voting systems trust

When the votes are collected and the results are to be summarized, k out of n shares of officers in charge is needed to proof correctness. This is required also to give confidence to the voters that there is no possible results interpretation.

3.3 Medical agreement

Some medical results are very sensitive and needs to be observed by several medical specialists together. This scenario is best served by at least k shares from a total of n shares to reconstruct the target-key TK to open the medical testing results. It can also be useful in securing information related to deep neural network based on sparse auto-encoder for voxel-wise detection of cerebral micro-bleed as detailed in [22].

3.4 Wills and inheritance

Normally the inheritance and wills letter can be reason for problems between people involved. A good way to reduce this problem and increase trust is to put the inheritance and wills letter into secret form that cannot be observed except by availability of most involved people. Secret sharing can help by requiring at least k from n shares to be available to reconstruct the target-key TK to open and study the inheritance and wills letter.

4 Proposed counting-based secret sharing technique

The proposed secret sharing method works on constructing a set of secret shares (A) from the Target Key TK where only selected (n) secret-shares can be useful, i.e. to our technique. Note that subset $n \in A$ is the useful shares to be distributed among application participants. Based on the user application requirements (k) shares are chosen, I mean k out of n shares, where these k are selected sufficient to be combined to reconstruct the Target Key TK. This process of combining the k shares to reconstruct TK is performed through our specific Counting-Based method. Our complete proposed counting-based secret sharing technique process can be observed within the analogy of Fig. 1.

The proposal process covers the method of generating the shares as well as the combining reconstruction scheme generating the Target Key *TK*. Observe Fig. 1, two options have been presented for providing the secret shares collection *A* where the subset *n* of useful shares can be selected from. The two options are named as *1-bit* method or *2-bits* method for secret shares generation, where both are elaborated later showing their pros and cons. Consider the *k* selection where $k \in n$ shares are needed as input to the counting-based combination technique to regenerate the target key *TK*. As in Fig. 1, our counting-based reconstruction technique is exactly same for both *1-bit* or *2-bits* secret shares methods.

Note that generating the shares sizes are performed assuming that *k* is of same size as the Target Key *TK* to insure appropriate security reducing guessing probability of *TK* from shares. It is implicit that *TK* cannot be regenerated with less than *k* shares. It is to be mentioned that any $k \in n$ shares can be used together as input to the combination technique inputted in parallel to reconstruct *TK*. The proposed technique uses our counting-based method to add up the one-bits in the same position. If the counted bits are added up to the value of *k* (as the threshold), then the resulted output bit is assumed 1 otherwise the resulted bit is zero. These resulted bits are combined to construct the Target Key *TK* needed. The proposed counting-based secret sharing method can be formulated as Algorithm-1 shown below.

Algorithm-1:

1: Consider the zeros (zero bits) within the Target Key (*TK*)
2: Every zero in *TK* is changed to one giving different options of possible Shares
3: Choosing the suitable shares is based on the combination of shares and value of *k*
4: Select *k* Shares
5: Consider the selected *k* Shares in parallel
6: Count the ones showing its digital values m_x, where *x* is the position of bits within Shares
7: if $m_x > k\text{-}1$ represent this bit in position *x* with one, otherwise represent it zero.
8: Compare the represented bits with *TK*
9: if represented bits = *TK* then shares are selected correct, GOTO step 13
10: else, change any selected share and GOTO step 5
11: if all selected shares are chosen then
12: Report: cannot construct secret sharing for this *TK* with *k*
13: End

The algorithm is modeled and simulated as described in Section 5. To clarify the technique algorithm concept in depth, a clarification example is detailed next followed by general examples to proof the concept more. The clarification example covers several scenarios of choosing different values of *n* and its threshold choice *k*. The example also proves the applicability of the technique to stand against the security intruders attempts.

All the different examples are considering the proposed counting-based scheme in binary format. However, the numbers are also given in hexadecimal (Hex) numbering for clarification purposes. Hex numbers are to simplify tracing the results to avoid typo errors and confusions. Providing binary as well as hexadecimal digits are to also show the user's possible view of the numbers. We stress that what the user is going to observe cannot be related straightforwardly to

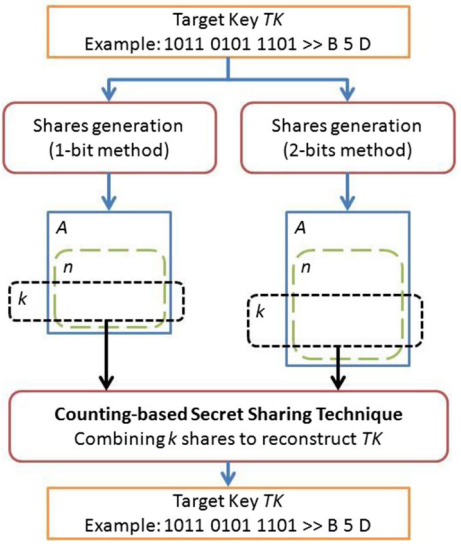

Fig. 1 Proposed counting-based secret sharing process

the shares nor to the secret target key *TK*, which are simply verified via hexadecimal numbers easier than binary. In fact, not using Hexadecimal makes the readers unsure, where using Hex numbers are also preferred by many related researchers making them prefer it to follow the work and its objective in clean clear smooth flow [10].

4.1 Clarification example

To describe our proposed counting-based secret sharing technique a simplified example is used. This example is not focusing on the shares generation, i.e. how to develop n or k shares, which will be presented later in the next section as *1-bit* or *2-bits* shares generation methods. The focus of this example is to show the counting-based technique recombination to generate *TK*. The example will study the condition of assuming $k = 4$ combining the shares in six conditions. It will first show the two conditions to get back *TK* in a very formal correct circumstance as well as the scenario condition of combining shares > k. Then, the example will explain the four false (incorrect *TK* output) conditions of combining shares < k, involving one false (intruder) share, involving several false (intruder) shares, and condition of combining repeated shares. The assumption is studying the secret *Target Key TK* as: 1011 0101 1101; supposing $n = 8$ and $k = 4$.

This *Target Key*:	1011 0101 1101	>>	B 5 D (in Hexadecimal)
Then the proposed n shares extracted from our scheme are as follows:			
Share 1:	1011 0101 1111	>>	B 5 F
Share 2:	1011 0111 1101	>>	B 7 D
Share 3:	1011 1101 1101	>>	B D D
Share 4:	1111 0101 1101	>>	F 5 D
Share 5:	1011 0111 1111	>>	B 7 F
Share 6:	1011 1111 1101	>>	B F D
Share 7:	1111 1101 1101	>>	F D D
Share 8:	1111 0101 1111	>>	F 5 F

Note that many other unsuitable shares can be listed as part of A but eliminated from n; since n will consider the shares that can be useful to be chosen as k. To be brief, not all

possible shares of A are suitable, only the selected n shares are elected such that to work correctly in the proposed counting-based technique recombination to generate TK as clarified in the following cases:

Case 1: Condition of Combining Shares = k

Observing the $n = 8$ shares above, any k ($k = 4$ in this example case) shares are combined and their position counting of one-bits are computed as the following:

Share 2:	1011 0111 1101	>>	B 7 D
Share 3:	1011 1101 1101	>>	B D D
Share 6:	1011 1111 1101	>>	B F D
Share 8:	1111 0101 1111	>>	F 5 F
Counting result:	4144 2424 4414		

In the positions with counting result = k, i.e. counting bits = 4 in our case, that position is given a one, otherwise, a zero is placed in that location.

So…

Counting results:	4144 2424 4414		
Output =	1011 0101 1101	>>	B 5 D

Case 2: Condition of Combining Shares > k

This system is valid even if more than k shares are combined with condition that the counting result is equal to or greater than k, i.e. counting result $\geq k$, such as the following:

Share 1:	1011 0101 1111	>>	B 5 F
Share 3:	1011 1101 1101	>>	B D D
Share 4:	1111 0101 1101	>>	F 5 D
Share 5:	1011 0111 1111	>>	B 7 F
Share 7:	1111 1101 1101	>>	F D D
Share 8:	1111 0101 1111	>>	F 5 F
Counting results:	6366 2616 6636		
Output:	1011 0101 1101	>>	B 5 D

So the condition to get the results back is to count the one-bits within the same location of the shares, whenever the counting result $\geq k$ the output is one; otherwise the counting result $< k$ and the output is zero.

Case 3: Condition of Combining Shares < k

Assume the shares available are less than k such as the example below:

Share 8:	1111 0101 1111	>>	F 5 F
Share 7:	1111 1101 1101	>>	F D D
Share 4:	1111 0101 1101	>>	F 5 D
Counting results:	3333 1303 3313		
Output:	0000 0000 0000	>>	0 0 0

The output obtained is zero, i.e. not correct, assuring that our proposed method is correct.

Case 4: Condition of Combining Shares and one false (intruder) share

Assume the shares available are less than k combined with one intruder false share such as the example shown below:

Share 1:	1011 0101 1111	>>	B 5 F	
Share 2:	1011 0111 1101	>>	B 7 D	
Share 3:	1011 1101 1101	>>	B D D	
False Share:	0101 0011 1111	>>	5 3 F	(Intruder Share)
Counting results:	3134 1324 4424			
Output:	0001 0001 1101	>>	1 1 D	

Case 5: **Condition of Combining Shares and several false (intruder) shares**

Assume the shares available are combined with several intruder false shares such as the example shown below:

Share 4:	1111 0101 1101	>>	F 5 D	
Share 5:	1011 0111 1111	>>	B 7 F	
Share 6:	1011 1111 1101	>>	B F D	
False Share:	0101 1001 1111	>>	5 9 F	(Intruder Share)
False Share:	0100 1100 1010	>>	4 C A	(Intruder Share)
False Share:	0101 1111 0110	>>	5 F 6	(Intruder Share)
Counting results:	3435 4535 5544			
Output:	0101 1101 1111	>>	5 D F	(wrong result as expected)

The result shows completely wrong results which is expected.

Case 6: Condition of Combining Repeated Shares

Share 4:	1111 0101 1101	>>	F 5 D	
Share 7:	1111 1101 1101	>>	F D D	
Share 8:	1111 0101 1111	>>	F 5 F	
Counting results:	4444 1404 4424			
Output:	1111 0101 1101	>>	F 5 F	(wrong result as expected)

The result shows completely wrong result which confirms the study expectation.

4.2 General examples as proof of concept

This general examples subsection presents different cases to clarify and compare the idea assuming simple different *Target Key (TK)* sizes. The study involved scenarios of

variations of *TK*, i.e. *TK = 4-bits*, *TK = 5-bits*, *TK = 6-bits*, *TK = 8bits*, and *TK = 12-bits*, where the simplicity is only for describing the concept and cannot be used in real-life applications. Real-life examples need more large numbers involving minimum *TK = 64-bits* as password numbers involving 8 digits.

4.2.1 General Example 1 (TK = 4-Bits)

Consider the simple example of *TK = 4-bits*, with *n = 4* shares, and *k* = 2 shares, where 2-keys are to be sufficient for retrieving *TK*. Observe the combination and their position counting of one-bits, which are computed as the following:

Target Key TK: 0011 >> 3 (in Hexadecimal)
Then, the proposed *n* shares extracted from our scheme can be as follows:

Share 1:	1011	>>	9
Share 2:	0111	>>	7
Share 3:	1111	>>	F
Share 4:	1111	>>	F (Similar to Share 3)

Assuming any two shares to be used, for example, Shares 1 & 2 :

Share 1:	1011	>>	9
Share 2:	0111	>>	7
Counting result:	1122		
Output =	0011	>>	3 (As *TK* needed)

4.2.2 General Example 2 (TK = 5-Bits)

Assume the straightforward example of *TK = 5-bits*, with *n = 6* shares, and *k* = 6 shares, such that all *n = 6-keys* are to be used for retrieving *TK*. The combination and their position counting of one-bits, which are detailed as the following:

Target Key TK: 00110 >> 06 (in Hexadecimal)
The proposed *n* shares are the same *k* shares (in this case) related to *TK* as follows:

Share 1:	10110	>>	16
Share 2:	01110	>>	0E
Share 3:	00111	>>	07
Share 4:	11110	>>	1E
Share 5:	10111	>>	17
Share 6:	01111	>>	0F
Counting result:	33663		
Output =	00110	>>	03 (As *TK* required)

4.2.3 General example 3 (TK = 6-Bits)

Consider the general example of *TK = 6-bits*, with *n = 8* shares, and *k* = 2 shares. Observe the combination and their position counting of one-bits, which are studied as the following:

Target Key TK: 10 0101 >> 25

Then, the proposed *n* shares extracted from our scheme can be as follows:

Share 1: 11 0101 >> 35
Share 2: 10 0111 >> 27
Share 3: 10 1101 >> 2D
Share 4: 11 1101 >> 3D
Share 5: 11 0111 >> 37

To fulfill *k=2* requirement, any two shares are selected, such as Shares 1 & 2 :

Share 1: 11 0101 >> 35
Share 2: 10 0111 >> 27

Counting result: 21 0202
Output = 10 0101 >> 25 (As *TK* required)

Assuming the two shares to be used are Shares 3 & 4, gives error as below:

Share 3: 10 1101 >> 2D
Share 4: 11 1101 >> 3D

Counting result: 21 2202
Output = 10 1101 >> 2D (Wrong *answer* insisting on confirmation requirement, such that not all shares can be combined together to generate the needed *TK*)

Involving Share 3 & 5, as any two shares scenario example, shows mistake, as follows:

Share 1: 11 0101 >> 35
Share 5: 11 0111 >> 37

Counting result: 22 0212
Output = 11 0101 >> 35 (error *answer* confirming previous situation)

4.2.4 General Example 4 (TK = 8-Bits)

Observe an example of *TK = 8-bits*, with *n = 8* shares, and *k* = 5 shares, observe the combination and their position counting of one-bits, which are computed as the following:

Target Key TK: 1000 0101 >> 85 (in Hexadecimal)

Then, the proposed n shares extracted from our scheme can be as follows:

Share 1: 1100 0101 >> C5
Share 2: 1010 0101 >> A5
Share 3: 1001 0101 >> 95
Share 4: 1000 1101 >> 8D
Share 5: 1000 0111 >> 87
Share 6: 1110 0101 >> E5
Share 7: 1101 0101 >> D5
Share 8: 1100 1101 >> CD
Share 9: 1100 0111 >> C7
Share 10: 1011 0101 >> B5
Share 11: 1010 1101 >> AD
Share 12: 1010 0111 >> A7
Share 13: 1001 1101 >> 9D
Share 14: 1001 0111 >> 97
Share 15: 1000 1111 >> 8F

The first *n* shares, i.e. for k = 5, are assumed to be used from our scheme can be as follows:

Share 1:	1100 0101	>>	C5	
Share 2:	1010 0101	>>	A5	
Share 3:	1001 0101	>>	95	
Share 4:	1000 1101	>>	8D	
Share 5:	1000 0111	>>	87	
Counting result:	5111 1515			
Output =	1000 0101	>>	85	(As *TK* required)

Assume different *n* shares, i.e. Shares: 6,7,8,9,10 are to be used as follows:

Share 6:	1110 0101	>>	E5
Share 7:	1101 0101	>>	D5
Share 8:	1100 1101	>>	CD
Share 9:	1100 0111	>>	C7
Share 10:	1011 0101	>>	B5
Counting result:	5422 1515		
Output =	1000 0101	>>	85

Note that the output only considered counting results ≥ *k* (≥5 in this case) getting correct answer.

Suppose the *n* shares selected are: Shares: 11,12,13,14,15, to be involved as follows:

Share 11:	1010 1101	>>	AD	
Share 12:	1010 0111	>>	A7	
Share 13:	1001 1101	>>	9D	
Share 14:	1001 0111	>>	97	
Share 15:	1000 1111	>>	8F	
Counting result:	5023 3535			
Output =	1000 0101	>>	85	(As *TK* needed)

4.2.5 General Example 5 (TK = 8-Bits)

Another complex example of *TK = 8-bits*, with *n = 8* shares, and *k* = 4 shares, observe the combination and their position counting of one-bits, which are computed as the following:

Target Key TK: 1001 0010 >> 92 (in Hexadecimal)

Then the proposed n shares extracted from our scheme are as follows:

Share 1:	1101 0010	>>	D2	
Share 2:	1011 0010	>>	B2	
Share 3:	1001 1010	>>	9A	
Share 4:	1001 0110	>>	96	
Counting result:	4114 1140			
Output =	1001 0010	>>	92	(As needed)

Which proofed the condition to get the results back is to count the one-bits within the same location of the shares, whenever the counting result ≥k the output is one; otherwise the counting result <k and the output is zero.

Assume to have different shares in relation to the same *TK* as follows:

Share 5:	1001 0011	>>	93	
Share 6:	1111 0010	>>	F2	
Share 7:	1101 1010	>>	DA	
Share 8:	1101 0110	>>	D6	
Counting result:	43141141			
Output =	10010010	>>	92	(As Needed)

We can have different shares in relation to the same *TK* as follows:

Share 9:	1101 0011	>>	D3	
Share 10:	1011 1010	>>	BA	
Share 11:	1011 0110	>>	B6	
Share 12:	1011 0011	>>	b3	
Counting result:	41341142			
Output =	10010010	>>	92	(As Expected)

Consider this wrong scenario having fewer shares than needed in relation to the same *TK* as follows:

Share 13:	1001 1110	>>	9E
Share 14:	1001 1011	>>	9B
Share 15:	1001 0111	>>	97

Counting result: 30032232 : mistake verifying the needed case of number of keys less than k=4 >> Output = 00000000 >> 00 (Error *TK* as Required)

4.2.6 General Example 6 (TK = 12-Bits)

The final complex considered example is of *TK = 12-bits*, with $n = 8$ shares, and $k = 3$ shares, which is considered as a very simple real-life scenario. Observe the combination and their position counting of one-bits, which are presented as the following:

Target Key TK: 1011 0101 1101 >> B5D

Possible *n* shares generated from TK can be from the following:

Share 1:	1111 0101 1101	>>	F5D
Share 2:	1011 1101 1101	>>	BDD
Share 3:	1011 0111 1101	>>	B7D
Share 4:	1011 0101 1111	>>	B5F
Share 5:	1111 1101 1101	>>	FDD
Share 6:	1111 0111 1101	>>	F7D
Share 7:	1111 0101 1111	>>	F5F
Share 8:	1011 1111 1101	>>	BFD
Share 9:	1011 1101 1111	>>	BDF
Share 10:	1011 0111 1111	>>	B7F

Suppose the *n* shares selected are: Shares: 1,2,3, to be involved as follows:

Share 1:	1111 0101 1101	>>	F5D
Share 2:	1011 1101 1101	>>	BDD
Share 3:	1011 0111 1101	>>	B7D
Counting result:	3133 1313 3303		
Output =	1011 0101 1101	>>	B5D (As *TK* Needed)

We can have different shares: 5,6,7, in relation to the same *TK* but giving error as follows:

Share 5:	1111 1101 1101	>>	FDD
Share 6:	1111 0111 1101	>>	F7D
Share 7:	1111 0101 1111	>>	F5F
Counting result:	3333 1313 3313		
Output =	1111 0101 1101	>>	F5D (error: wrong *answer* confirming verification essential requirement to proof the shares combination validity)

The concept is that not all shares can be combined to provide the correct needed *TK* output. Selected shares necessitate to be verified for its validity together, then to be provided to the application for usage. A final correct shares selection scenario is given below:

Share 5:	1111 1101 1101	>>	FDD
Share 6:	1111 0111 1101	>>	F7D
Share 3:	1011 0111 1101	>>	B7D
Counting result:	3233 1323 3303		
Output =	1011 0101 1101	>>	B5D (As *TK* Needed)

5 Modeling & simulation

This section will present modeling and simulating of our counting-based secret sharing technique process. We will be covering the methods of generating the useful shares as well as the combining reconstruction scheme to generate the *Target Key TK*. The simulation platform used is MATLAB as briefed in next subsection. Then, the reconstruction scheme to generate the *TK* will be discussed. After that, the options of procedures for providing the appropriate secret shares *n* are presented, i.e. elaborating on the 1-bit method as well as the 2-bits method briefing their pros and cons.

5.1 Simulation platform

The proposed technique and its shares generation are all modeled via MATLAB on an Intel processor Core i7 PC of speed 2.39 GHz, Ram 8 GB, 64-bit operating system. The MATLAB code to reconstruct the target key as well as the code to generate the secret shares has been implemented in details using the available MATLAB version R2015a platform. MATLAB programming is used because of its simplicity, high-performance numerical computation, common data analysis, applicable visualization capabilities, and accessible application development tools. In addition, MATLAB functions are

academically known easy to use; where a user interface function guides through the process. MATLAB also provides the ability to call external libraries, such as Open CV providing immediate access to thousands of fundamental and specialty functions written by experts. MATLAB has large user community with lots of open-access codes for knowledge sharing. Furthermore, all features found in MATLAB software platform can be documented clearly and elaborated with many examples.

5.2 Modeling counting-based secret sharing scheme to generate the target key

The proposed technique algorithm, as presented earlier in section 4, is implemented using MATLAB. The code is focusing on reconstructing the target key *TK*. The technique algorithm is modeled as the data-flow diagram shown in Fig. 2. The scheme interface is shown in Fig. 3 for the case of $k = 2$ for simplicity.

5.3 Modeling the secret shares generation

The proposed secret sharing process (as Fig. 1) requires appropriate procedures for the *n* shares generation. All shares possibilities *A* can be easily offered where only the selected *n* shares are useful to our scheme, as discussed earlier. We propose two approaches for constructing the acceptable shares *n* for this counting-based secret sharing technique. Other different procedures for generating *n* shares can be thought of which are kept as open research studies to be followed as future work. The presented procedures options for providing the appropriate secret shares *n* are named as the 1-bit method as well as the 2-bits method. Both methods are modeled and simulated via MATLAB as follows.

5.3.1 Secret shares generation via 1-bit method

The *n* secret shares generation can be performed through this simple 1-bit method. The method is mainly considering the zero bits within the target key *TK*. Every share is selecting only one zero of *TK* flipping it to one producing it as a valid secret share. So every share of *n* is basically the target key *TK* but with one of its zero's turned to one. This procedure, i.e. 1-bit method, is very simple to implement, fast to run, confirmed reliable, but limited in number of shares produced. I mean the value of *n* cannot be more than the number of zeros found within *TK* making it tight and data dependant. As the number of zeros within *TK* reduces, the number of *n* shares using this 1-bit method decreases. The method is formulated as data-flow model shown in Fig. 4. The implementation interface of the study MATLAB simulation is shown in Fig. 5 where a simplified example of *TK* = [1 0 0 1 0 0 1 0] is presented as below:

TK:	1 0 0 1 0 0 1 0	>>	92 Hex
Share 1:	1 <u>1</u> 0 1 0 0 1 0	>>	D2
Share 2:	1 0 <u>1</u> 1 0 0 1 0	>>	B2
Share 3:	1 0 0 1 <u>1</u> 0 1 0	>>	9A
Share 4:	1 0 0 1 0 <u>1</u> 1 0	>>	96
Share 5:	1 0 0 1 0 0 1 <u>1</u>	>>	93

The underlined ones, i.e. within the example shares, are the changed zeros from *TK*. Note that this example of 1-bit shares generation method limited to number of zeros within *TK* is having $n = 5$ although the set of all possibilities of *A* can go theoretically to almost 2^8–1.

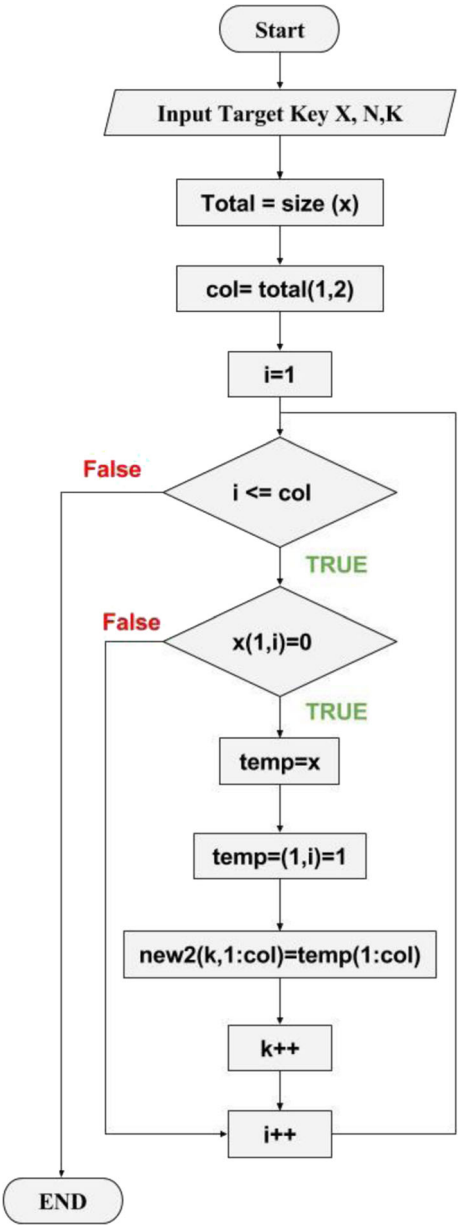

Fig. 2 Modeling the scheme for *TK* reconstruction

5.3.2 Secret shares generation via 2-bits method

The proposed 2-bits method for shares generation is built upon the 1-bit method as an extension. It is considering two-zeros after completing the 1-bit method one-zeros shares generation. The method begins by scanning the zeros within *TK* flipping them to ones as the 1-bit method followed by flipping 2-zeros per share generating applicable more shares. The only concern providing these more shares than previous 1-bit ones is its essential need for checking the values to be appropriate, i.e. useful shares before

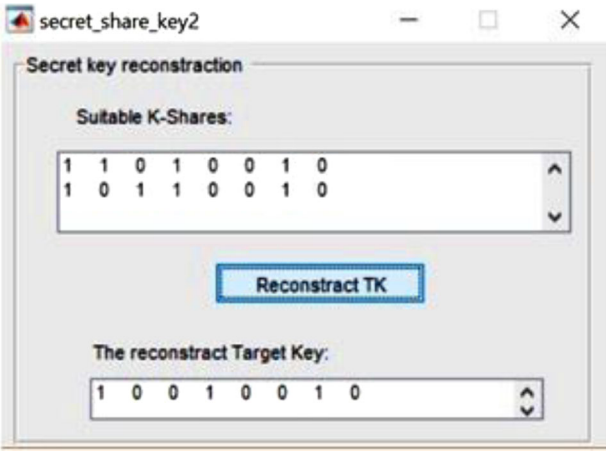

Fig. 3 Implementation interface for *TK* reconstruction scheme

providing them to the participants. Not all the produced shares are practical for our counting-based *TK* reconstruction scheme. This *n* shares verification needs to be performed for the 2-bits method checking its applicability (to be part of *n*) which is not needed for the 1-bit method, as a complexity price paid to gain more shares. The method is formulated as data-flow model shown in Fig. 6.

It is to be mentioned that other shares can be guessed and found working fine in some scenarios. However, these guessed shares cannot be confirmed appropriate for all scenarios of our counting-based secret sharing technique making avoiding them, i.e. for consistency of this 2-bits shares generation method.

To clarify the concept, the same simplified example of *TK* = [1 0 0 1 0 0 1 0] is used for the 2-bits method as presented below:

TK:	1 0 0 1 0 0 1 0	>>	92 Hex
Share 1:	1 1 0 1 0 0 1 0	>>	D2
Share 2:	1 0 1 1 0 0 1 0	>>	B2
Share 3:	1 0 0 1 1 0 1 0	>>	9A
Share 4:	1 0 0 1 0 1 1 0	>>	96
Share 5:	1 0 0 1 0 0 1 1	>>	93
Share 6:	1 1 1 1 0 0 1 0	>>	F2
Share 7:	1 1 0 1 1 0 1 0	>>	DA
Share 8:	1 1 0 1 0 1 1 0	>>	D6
Share 9:	1 1 0 1 0 0 1 1	>>	D3
Share 10:	1 0 1 1 1 0 1 0	>>	BA
Share 11:	1 0 1 1 0 1 1 0	>>	B6
Share 12:	1 0 1 1 0 0 1 1	>>	B3
Share 13:	1 0 0 1 1 1 1 0	>>	9E
Share 14:	1 0 0 1 1 0 1 1	>>	9B
Share 15:	1 0 0 1 0 1 1 1	>>	97

The shares generated in the example above can be observed as shares 1–5 similar to result of the 1-bit method and shares 6–15 as the extension, making them all shares as output of 2-bits method. These shares generated cannot be used immediately when selecting *k*. The shares need to be tested

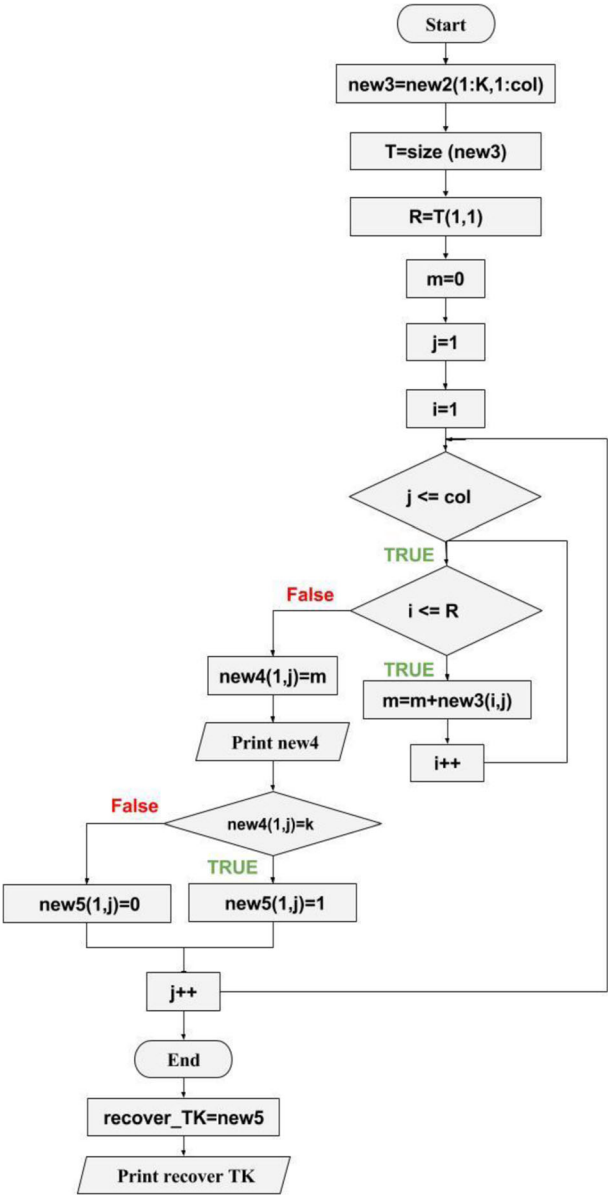

Fig. 4 Modeling the 1-bit shares generation method

for applicability, as discussed earlier, before using them. I mean the selection of n shares can change based on deciding the value of k. For our example, this testing of applicable shares to be used is found essential for $k < 6$. For example, assume k = 5, some useful sets of n shares to be used are:

$n = [1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13]; n = [1, 2, 3, 4, 5, 10, 11, 12, 13, 14, 15]; n = [1, 2, 3, 4, 5, 7, 8, 9, 10, 14, 15];$
$n = [1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 14, 15]; n = [1, 2, 3, 4, 6, 8, 9, 12, 13, 14, 15]; n = [2, 3, 4, 7, 8, 9, 10, 12, 14, 15]$

The selection test of appropriate shares can be achieved by parallel observation of the flipped bits of ones, i.e. the underlined ones within shares, counting their value to be less than

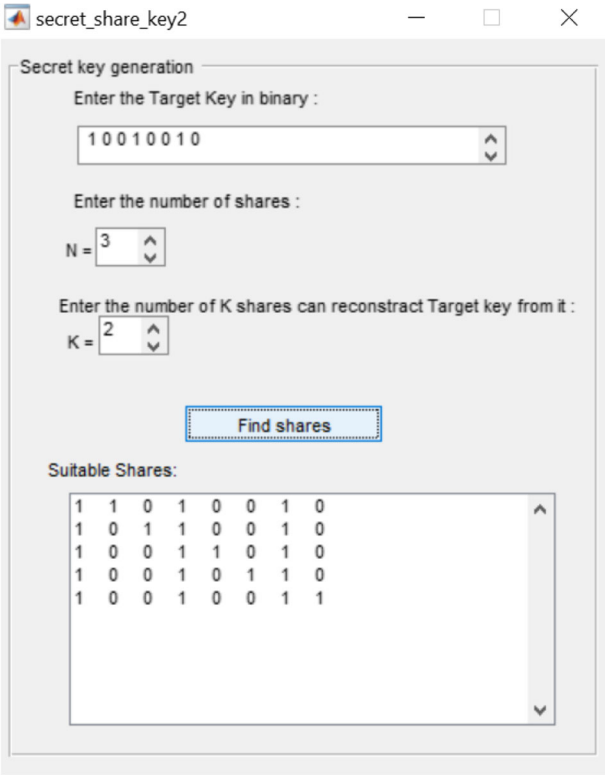

Fig. 5 Implementation interface of running 1-bit shares generation method

k. This counting examination is consistent to the philosophy of our proposed counting-based secret sharing technique.

6 Comparison & analysis

The study implementation elaborated on the two methods of secret shares generation, i.e. 1-bit method and 2-bits method, considered for a range of *Target Keys TK* starting from 00 0000 (00 Hex) to 11 1111 (FF Hex). The study also considered security level of the chosen *TK* in relation to the possible shares generated. All the possible shares for *TK* [00 0000 (00 Hex) to 11 1111 (FF Hex)] generated by the software programmed simulation are summarized in Table 1. This list is further studied in this section for comparison and analysis. For example, consider in Table 1 *TK = 10 0101* (25 Hex) can have only 3 possibilities using 1-bits method and 6 possibilities using 2-bits method. These shares possibilities are presented below:

TK:	1 0 0 1 0 1	>>	25 Hex
Share 1:	1 1 0 1 0 1	>>	35
Share 2:	1 0 1 1 0 1	>>	2D
Share 3:	1 0 0 1 0 1	>>	25
Share 4:	1 1 1 1 0 1	>>	3D
Share 5:	1 1 0 1 1 1	>>	37
Share 6:	1 0 1 1 1 1	>>	2F

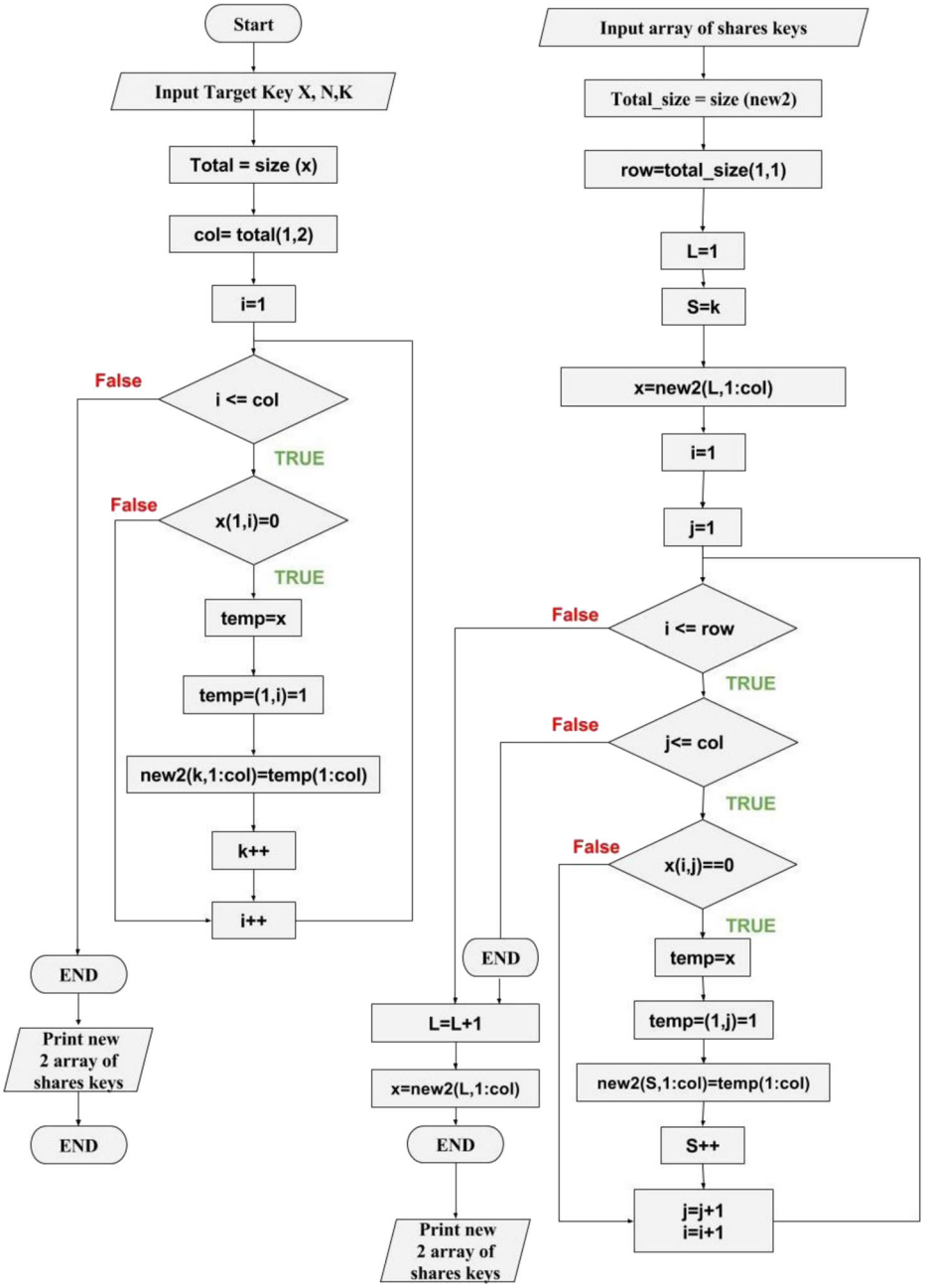

Fig. 6 Modeling the 2-bit shares generation method

It is observed clearly from the example of *TK = 10 0101* (25 Hex) which is consistent to all the listed results in Table 1 that the secret shares generation in both methods depend exclusively on the available zeros within *TK*. This zero dependency within *TK* is expected in accordance to the main idea behind this proposed counting based secret

Table 1 Listing number of shares possibilities generated by proposed two methods

Target key (Hexadecimal)	No. of shares 1 bit method	No. of shares 2 bit method
0	6	21
1	5	15
2	5	15
3	4	10
4	5	15
5	4	10
6	4	10
7	3	6
8	5	15
9	4	10
A	4	10
B	3	6
C	4	10
D	3	6
E	3	6
F	2	3
10	5	15
11	4	10
12	4	10
13	3	6
14	4	10
15	3	6
16	3	6
17	2	3
18	4	10
19	3	6
1A	3	6
1B	2	3
1C	3	6
1D	2	3
1E	2	3
1F	1 * error	1 * error
20	5	15
21	4	10
22	4	10
23	3	6
24	4	10
25	3	6
26	3	6
27	2	3
28	4	10
29	3	6
2A	3	6
2B	2	3
2C	3	6
2D	2	3
2E	2	3
2F	1 * error	1 * error
30	4	10
31	3	6
32	3	6
33	2	3
34	3	6
35	2	3
36	2	3

Table 1 (continued)

Target key (Hexadecimal)	No. of shares 1 bit method	No. of shares 2 bit method
37	1 * error	1 * error
38	3	6
39	2	3
3A	2	3
3B	1 * error	1 * error
3C	2	3
3D	1 * error	1 * error
3E	1 * error	1 * error
3F	0 * error	0 * error

sharing scheme. As the number of zeros increase, the shares to be generated get more providing options with increasing security to be utilized.

Some values in Table 1 shows the results "*error" in relation to specific *TK* values, such as: *TK = 1F, 2F, 37, 3B, 3D, 3E, 3F*. This error is declared due to the low number of zeros within *TK* content, i.e. just containing 1 bit of value zero or no zeros at all as in (3F). All the *TK* values with one zero or non are to be avoided because of the limitation in the shares generated according to this proposed counting method. Note that the one zero *TK* is just allowing one possible share which cannot be allowed in this system. The reconstruction of the *TK* back is trivially unsecure. For example, consider *TK = 0 1 1 1 1 1* (1F Hex). The possible share can just be: 1 1 1 1 1 1 (3F Hex); which is also the share found for *TK = 2F, 37, 3B, 3D, 3E*. The *TK* = 3F Hex is not acceptable since no shares can be generated for it in this proposed system. To summarize, the system cannot accept *TK* with low number of zeros. The rule of applicable *TK* is that (*TK* number of zeros must be greater than 1) for the shares generation scheme to work with minimum security. The security level increase as the number of zeros in *TK* gets to be more as will be detailed next.

Note that the number of possible shares, as shown in Figs. 7 and 8, varies independent to the changing of *TK* value nor consistent to its homogenous increasing. It is based on the data-dependant zeros in *TK* building interesting security randomization view, which is considered enhancing the security of the proposed system.

Consider Fig. 7 for 1-bit method, the security of the systems can be classified based on the number of possible shares to high-security, medium- security, and low- security. As can be observed in Fig. 7, the *TK* alternatives are divided according to the number of shares pointing to the suggested selected *TK* values. In this discussion of *TK* ranging: *00-3F Hex*, 1-bit method can choose high-security class within the *TK* having possible number of shares value 5 or more, such as *TK = 0, 1, 2, 4, 8, 10, 20 Hex*. Medium-security class of 1-bit method can be from selecting *TK* of possible shares with values 4 as *TK = 3, 5, 6, 9, A, C, 11, 12, 14, 18, 21, 22, 24, 28, 30 Hex*. Low-security class can be for the TK values of 3 or less, such as *TK = 7, B, D, E, F, 13, 15, 16, 17, 19, 1A, 1B, 1C, 1 D, 1E, 23, 25, 26, 27, 29, 2A, 2B, 2C, 2D, 2E, 31, 32, 33, 35, 36, 38, 39, 3A, 3C Hex*.

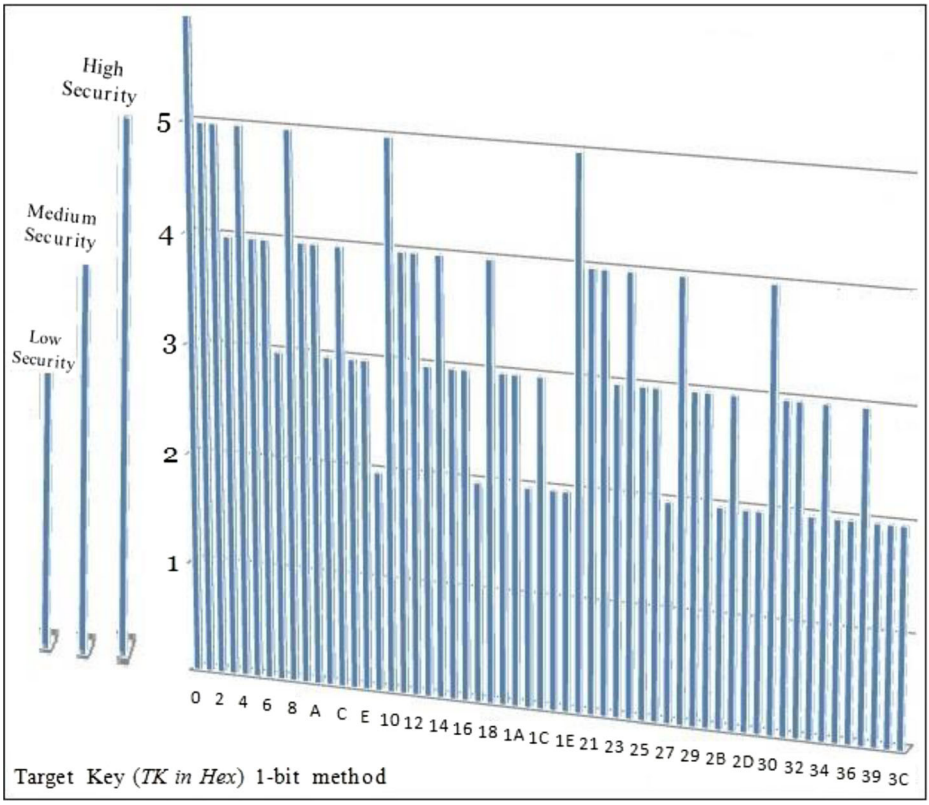

Fig. 7 Number of possible shares vs. Security for 1-bit method in relation to *TK* value

All these low-security *TK* values are considered inappropriate to be chosen for safe applications, although they can work technically fine running the proposed counting based secret sharing system.

Similarly, 2-bits method can categorize the *TK* values to high- security, medium-security, and low- security selections, as observed in Fig. 8. High- security selection recommend the *TK* having possible number of shares value not less than 15, such as *TK = 0, 1, 2, 4, 8, 10, 20 Hex*. Medium-security class of 2-bits method can be from selecting *TK* of possible shares with values 10 as *TK = 3, 5, 6, 9, A, C, 11, 12, 14, 18, 21, 22, 24, 28, 30 Hex*. Low-security class can be for the TK values of 6 or less, such as *TK = 7, B, D, E, F, 13, 15, 16, 17, 19, 1A, 1B, 1C, 1D, 1E, 23, 25, 26, 27, 29, 2A, 2B, 2C, 2D, 2E, 31, 32, 33, 35, 36, 38, 39, 3A, 3C Hex*. All these low-security *TK* values are considered unsuitable to be used for safe applications, although they can work algorithmically perfect, similar to the 1-bit method, theoretically activating the proposed counting based secret sharing system.

Interestingly the categorization of security levels of the 1-bit method and the 2-bits method are found closely associated, as can be seen in Fig. 9. In other words, the same *TK* selections for high-security, medium-security, and low-security are found consistent within the 1-bit method and the 2-bits method, allowing the generalization of the selection to be verified acceptable.

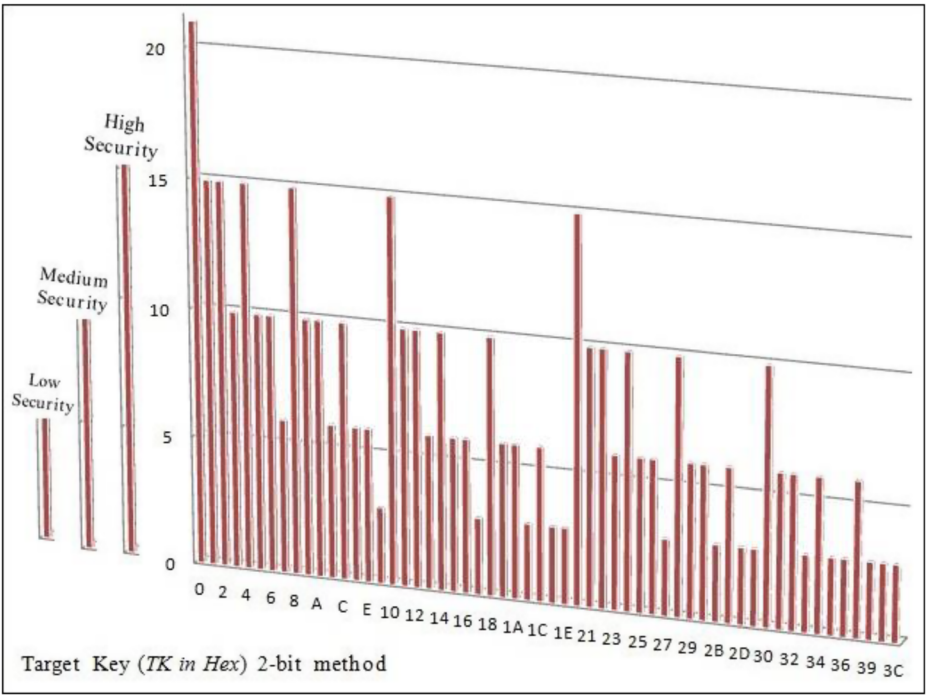

Fig. 8 Number of possible shares vs. Security for 2-bits method in relation to *TK* value

Statistically, considering the *TK* range *00-3F Hex* providing the number of possibilities that can be expected to form the entire range of 56 options, however, the reality is different. As shown in Fig. 9, the reality gives low number of possibilities based on the *TK* valid alternatives. This number of possibilities for every target key *TK* case is directly

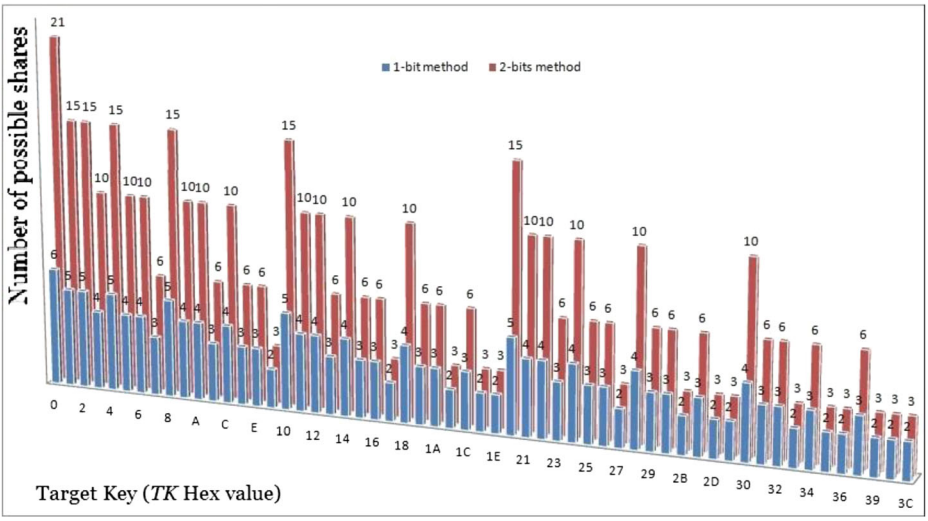

Fig. 9 Comparing the number of possible shares of every method in relation to *TK* value

proportional to the security level of the system. For example, the number of high-security *TK* possibilities is found to be 7 estimated as 12.5% of the total possible *TK* options making-up the assumption of high security percentage. The medium-security secret *TK* selections are found 15 alternatives out of 56, considered as 27% of the whole feasible *TK*. Finally, as obvious, the low-security *TK* pool is the largest. It is found allowing 34 possibilities out of 56, which is considered as 60.5% of the entire options. These statistical percentages values can be representative to real life applications as the number of bits of *TK* increase. The recommendation from this security level statistical analysis is to avoid working below high-security level, enforcing the real life applications to choose among large numbers.

7 Conclusion

This work proposed and implemented a new secret-sharing system that works based on parallel counting of the ones within the shares to generate the secret target key output. The work developed two different modeling variations for the secret shares generation, 1-bit method and the 2-bits method, where both are studied clearing their advantages and disadvantages. This counting-based secret sharing key study detailed the trade-off for selection process of number of secret shares vs. the target key based on the security level required by the application. Interestingly the study gave detailed examples to stress the differences in the target key preferences based on high, medium, and low security levels as needed according to the possible number of shares applicable to be used.

The three classification security levels of the 1-bit method and the 2-bits method are found data dependant related to the content of *TK* and its involvement of number of zeros. The *TK* selection is given possibility to be based on the security level required, which is found consistent within the 1-bit method as well as the 2-bits method. The simulation elaboration focused on possibilities of *Target Keys TK* ranging from 00 0000 (00 Hex) to 11 1111 (FF Hex) resulting the three levels of security as 12.5% high-security, 27% medium-security, and 60.5% low-security, as statistically option percentages to be considered. Notes that as the security level increase, the percentage reduce dramatically, which is the main drawback or price to be considered and studied as future research.

The modeling and simulation results have been very attractive allowing further innovation considerations in the shares generation methods. The proposed secret sharing system is described in a way believed to be simple and practical. It is working as seed for novel research to come to show improvements and modify this counting based secret sharing scheme making it specifically geared to the different applications. The research work and results are believed to be very attractive as a base for further research in this direction of secret sharing techniques and applications.

Acknowledgments The authors would like to thank Umm Al-Qura University (UQU) for hosting this research. Appreciation is given to our Master program allowing the graduate student (Miss Nouf Al-Juaid) as co-author of this paper from Shaqra University to work this research with us performing the modeling, testing and all

simulations. Thanks to the Department of Information and Technical Services at the Custodian of the Two Holy Mosques Institute of the Hajj and Umrah Research for encouraging this teamwork. Thanks to the hosting department, i.e. Computer Engineering Department at the College of Computer & Information Systems, for motivating this wonderful cooperation guided within Umm Al-Qura University, Makkah, Saudi Arabia.

References

1. Ahmadoh E, Gutub A (2015) Utilization of two diacritics for Arabic text steganography to enhance performance. Lecture Notes on Information Theory 3(1):42–47
2. Asmuth C, Bloom J (1983) A modular approach to key safeguarding. IEEE Trans Inf Theory 29(2): 208–210
3. G.R. Blakley 1979 “Safeguarding cryptographic keys”, Proc. of 1979 AFIPS National Computer Conference 48:313–317
4. Blundo C, De Santis A, Gargano L, Vaccaro U (1993) Secret sharing schemes with veto capabilities. Proceedings of the First French-Israeli Workshop, Paris, France:19–21
5. Blundo C, Cresti A, De Santis A, Vaccaro U (1996) Fully dynamic secret sharing schemes. Theor Comput Sci 165:407–440
6. Gutub A (2007) Area flexible GF(2k) elliptic curve cryptography coprocessor. International Arab Journal of Information Technology (IAJIT) 4(1):1–10
7. Gutub A (2007) High speed hardware architecture to compute GF(p) Montgomery inversion with scalability features. IET (IEE) Proc Computers & Digital Techniques 1(4):389–396
8. Gutub A (2007) Efficient utilization of scalable multipliers in parallel to compute GF(p) elliptic curve cryptographic operations. Kuwait Journal of Science & Engineering (KJSE) 34(2):165–182
9. Gutub A (2010) Preference of efficient architectures for GF(p) elliptic curve crypto operations using multiple parallel multipliers. Int J Secur 4(4):46–63
10. Gutub A, Amin A (1999) An expandable Montgomery modular multiplication processor. Eleventh international conference on microelectronics, ICM'99, pages: 173–176, Kuwait
11. Gutub A, Tahhan H (2003) Improving cryptographic architectures by adopting efficient adders in their modular multiplication hardware. The 9th annual gulf internet symposium, Khobar, Saudi Arabia, October 13-15
12. Gutub A, Tenca A (2004) Efficient scalable VLSI architecture for Montgomery inversion in GF(p). Integr VLSI J 37(2):103–120
13. Herzberg A, Jarecki S, Krawczyk H, Yung M (1995) 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 339e–352
14. Krawczyk H (1993) Secret Sharing Made Short. Proceedings of the 13th Annual International Cryptology Conference (CRYPTO’ 93), Santa Barbara, California, USA, 22–26 August, pp 136–146
15. Laih C-S, Harn L, Lee J-Y, Hwang T (1990) “Dynamic threshold scheme based on the definition of cross-product in an N-dimensional linear space”, proceedings of advances in cryptology (CRYPTO’ 89). Lect Notes Comput Sci 435:286–298
16. McEliece RJ, Sarwate ADV (1981) On sharing secrets and reed-Solomon codes. Commun ACM 24(9): 583–584
17. Ogata W, Kurosawa K, Stinson DR (2006) Optimum secret sharing scheme secure against cheating. SIAM J Discret Math 20(1):79–95
18. Savas E, Naseer M, Gutub A, Koc C (2005) Efficient unified Montgomery inversion with multi-bit shifting. IEE Proceedings Computers and Digital Techniques 152(4):489–498
19. Shamir A (1979) How to share a secret. Commun ACM 22:612–613
20. Tassa T (2007) Hierarchical threshold secret sharing. J Cryptol 20(2):237–264
21. Wang K, Zou X, Sui Y (2009) A Multiple Secret Sharing Scheme based on Matrix Projection. Proc. of the 33rd Annual IEEE International Computer Software and Applications Conference, pp 400–405
22. Zhang Y-D, Zhang Y, Hou X-X, Chen H, Wang S-H (2017) Seven-layer deep neural network based on sparse autoencoder for voxelwise detection of cerebral microbleed. Multimed Tools Appl:1–18. https://doi.org/10.1007/s11042-017-4554-8

Adnan Gutub is currently working as Professor in Computer Engineering Department specialized in Information and Computer Security within Umm Al Qura University (UQU), Makkah -Saudi Arabia.

His experience was gained from his previous long-time work in Computer Engineering Department at King Fahd University of Petroleum and Minerals (KFUPM) in Dhahran, Saudi Arabia. He received his Ph.D. degree (2002) in Electrical & Computer Engineering from Oregon State University, USA. He had his BS in Electrical Engineering and MS in Computer Engineering both from KFUPM, Saudi Arabia.

Adnan's research interests involved optimizing, modeling, simulating, and synthesizing VLSI hardware for crypto and security computer arithmetic operations. He worked on designing efficient integrated circuits for the Montgomery inverse computation in different finite fields. He has some work in modeling architectures for RSA and elliptic curve crypto operations. His current interest in computer security also involved steganography such as image based steganography and Arabic text steganography.

In summer 2013, Adnan has been awarded 3-month visiting scholar grant in collaboration with Purdue University, West Lafayette, Indiana, USA. He had been involved in research of current studies related to Arabic Text Steganography in Data Security as well as Elliptic Curve Crypto Processor Designs. He then completed parts of this research work in summer 2015 visiting University of California Santa Barbra. He had been involving his work in discussion ideas and outcomes relating to them and their exploration within the information security field as overall ultimate research as well as opening-up new ideas with Crypto-Code (a focus research group at University of California Santa Barbra) making his specific scientific investigation internationally recognized.

Previously, Adnan have been twice awarded the UK visiting internship for 2 months of summer 2005 and summer 2008, both sponsored by the British Council in Saudi Arabia. The 2005 summer research visit was at Brunel University to collaborate with the Bio-Inspired Intelligent System (BIIS) research group in a project to speed-up a scalable modular inversion hardware architecture. The 2008 visit was at University of Southampton with the Pervasive Systems Centre (PSC) for research related to text steganography and data security.

Administratively, Prof. Adnan Gutub filled many executive and managerial academic positions at KFUPM as well as UQU. At KFUPM - Dhahran, he had the experience of chairing the Computer Engineering department (COE) for five years until moving to UQU - Makkah in 2010. Then, at UQU - Makkah, Adnan Chaired the Information Systems Department at the College of Computer & Information Systems followed by his leadership of the Center of Research Excellence in Hajj and Omrah (HajjCoRE) serving as HajjCoRE director for around 3-years until the end of 2013. Then, he was assigned his previous position as the Vice Dean of the Custodian of the Two Holy Mosques Institute of the Hajj & Omrah Research, within Umm Al Qura University (UQU), Makkah - Saudi Arabia.

Nouf Al-Juaid is a Teacher Assistant at Shaqra University, Shaqra, Saudi Arabia. She obtained Master of Sciences (MS) degree in Computer Sciences & Engineering, at Umm Al Qura University (UQU) fully sponsored by Shaqra University under the umbrella of Ministry of Higher Education. Her MS program at UQU is specialized in the information security track offered by the College of Computer and Information Systems offered at UQU-Makkah Campus, Saudi Arabia. In 2010, Nouf completed her Bachelor of Sciences (BS) degree with honors from Taif University Saudi Arabia. Nouf followed her BS studies by pursuing a higher diploma degree in education also from Taif University completed by the end of 2011. She, then, worked as official trainers at the Saudi Institute of Taif for around a year, i.e. until 2012, were she has been employed by Shaqra University as Graduate Teaching Assistant in the field of computing. At Shaqra, Nouf was assigned to teach introduction to computer science course classes as well as MATLAB classes based on her strong background and experience with programming languages such as MATLAB, Java, C++, PHP, and her outstanding ability to work with some databases like Oracle and SQL Nouf research capability started by her BS graduation project about multimedia medical records in radiology department using techniques of expert systems. Then, in her MS studies at UQU, she worked on building a program that is reconstructing permutations from differences sequence, which was a project related to the graduate course of analysis of algorithms. After, completing her tenure of her scholarship at UQU, she had been assigned the vice role of Computer Department Chairman at Shaqra University. Nouf Al-Juaid research interest focused on Computer and Information Security include Cryptography, Steganography, Cyber Security, Network Security, Networks, Artificial Intelligence, Image Processing, and Expert Systems.

Esam Khan is currently the vice dean of academic affairs at the Custodian of the Two Holy Mosques Institute for Hajj and Umrah Research, Umm Al-Qura University, Makkah, Saudi Arabia. He is an assistant professor in computer engineering. He received his B.Sc. in Computer Engineering (first class honor) in June 1999, and his M.Sc. in Computer Engineering in June 2001, both from the Department of Computer Engineering, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia. He received his PhD in Electrical and Computer Engineering in November 2005 from the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada. His M.Sc. thesis was about compression techniques of testing data. His Ph.D. dissertation was about hardware implementation of hash functions. His research interests include security and cryptography, and applications of information technology in Hajj and Umrah. He published several journal and conference papers in the areas of his research.