Journal of Information Security and Applications 60 (2021) 102869

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Collusion resistant secret sharing scheme for secure data storage and processing over cloud

Lakshmi V.S.[1,*], Deepthi S.[2], Deepthi P.P.

Department of Electronics and Communication Engineering, National Institute of Technology Calicut, Kerala, India

ARTICLE INFO

Keywords:
Data security
Cloud computing
Privacy preserving
Secret sharing
Homomorphic computation

ABSTRACT

Shamir secret sharing (SSS) is considered as a promising method for outsourcing the data securely due to its ability to support privacy preserving data processing while ensuring data availability. Major drawbacks of original SSS scheme are its susceptibility to collusion attack and high storage overhead. Hence in this paper, we first propose a modified SSS scheme (MSSS) which can resist collusion attack and provide adequate security even with two shares. However, the storage overhead of this scheme is high when it is extended to ensure data availability and integrity in cloud storage systems. Therefore, a modified ramp secret sharing (MRSS) with reduced storage overhead compared to MSSS scheme is also proposed in this paper. The proposed schemes can be employed for any privacy preserving data processing application which involve linear operations on the data. In this paper, in order to demonstrate the capability of proposed schemes to support privacy preserving data processing, Haar discrete wavelet transform (DWT) computation on medical images is considered as an example as DWT is widely used in feature extraction for disease diagnosis from pathological images. We present an algorithm for computing Haar DWT from medical image shares. The security of the proposed scheme is evaluated through mathematical cryptanalysis and resistance against various statistical attacks. The performance analysis shows that shared domain DWT offers same accuracy levels as that of plaintext domain.

1. Introduction

The advancements in cloud storage and computing paradigm has attracted various sectors due to the exponential growth in the data generation rate. Although cloud computing platform relieves its users from the burden of storage and processing cost, it also introduces several security threats such as availability, integrity and confidentiality [1–3]. In order to resolve the availability issue, cloud systems usually rely on distributed storage architecture, where the data is striped and stored redundantly among multiple servers using techniques such as replication and erasure codes [4,5]. On the other hand, detection of tampering of data and identification of corrupted servers are often achieved through integrity verification schemes which demands the storage of authentication tags along with the outsourced data [6]. Also, it is essential to ensure data privacy or confidentiality especially while outsourcing data of highly sensitive nature for storage and processing [7–10]. Even though traditional encryption schemes like advanced encryption standard (AES) provide data confidentiality, it will not support encrypted domain processing [11]. In order to facilitate operations in the encrypted domain (ED), the encryption scheme should be homomorphic to the computations performed on encrypted data [12].

Majority of the cloud processing tasks involve only linear operations and an additive homomorphic encryption scheme (HES) is sufficient for encrypting the data to be outsourced in such applications [13]. The most popular additive HES available in literature is Paillier scheme [14,15]. However, the major disadvantage of the Paillier scheme is high computational complexity due to exponentiation operations over large finite field [16]. Shamir secret sharing (SSS) [17,18] scheme has proved to be an effective additive homomorphic scheme, especially in privacy preserving data processing. Moreover, SSS has less computational complexity due to linear operations involved in generation and reconstruction of shares.

(t, n) SSS is a threshold based scheme in which n shares, generated from a secret, are distributed among n cloud service providers (CSP) and only t shares ($t \leq n$) are needed for reconstruction of the secret. Since size of each share is same as the size of the secret, the storage overhead increases with the number of shares being stored. However, SSS can provide data confidentiality, integrity and availability simultaneously with this storage overhead whereas Paillier scheme provides only data confidentiality. (t, n) SSS ensures data availability as the

* Corresponding author.
E-mail address: lakshmivs23@gmail.com (Lakshmi V.S.).
[1] Department of Electronics and Communication Engineering, Sree Chitra Thirunal College of Engineering, Trivandrum, Kerala, India.
[2] Wipro GE Healthcare Pvt. Ltd., Bangalore, India.

https://doi.org/10.1016/j.jisa.2021.102869

Available online 19 May 2021

secret can be reconstructed from any t out of n shares. Similarly, SSS provide data integrity verification by reconstructing the secret from two different sets of shares. Furthermore, SSS offers information theoretic security, which means that an adversary having infinite computational power cannot obtain any information about the secret even if he knows (t - 1) shares [17].

1.1. Related works

In this section, we briefly discuss the various secret sharing based schemes and their applications in homomorphic processing. Secret sharing was first introduced by Shamir [17] and Blakely [19] independently for safe management of cryptographic keys. Shamir (t, n) threshold secret sharing scheme is based on Lagrange polynomial interpolation whereas Blakley′s scheme is based on projective spaces. Both these schemes are perfect [20] as no information is leaked even if (t -1) participants try to retrieve the secret. These schemes are also ideal [21] since the information rate, which is the ratio of the size of the secrets to the size of shares is 1. This implies that the share size is same as secret. In order to reduce the computational complexity involved in reconstructing the secret, threshold schemes based on the Chinese Remainder Theorem (CRT) were proposed by Mignotte [22] and Asmuth-Bloom [23]. The problem with these schemes is the information leakage during collusion of t -1 participants depends on the parameters of the scheme and they are non-ideal. But Goldreich et al. [24] showed that if the parameters are consecutive primes, the security of threshold secret sharing schemes based on CRT can be improved. As an improvement of this result, Quisquater et al. [25] proved that these CRT based schemes can be made asymptotically ideal and perfect if the secret is uniformly chosen and parameters are consecutive primes. However, it is difficult to construct threshold schemes based on CRT since it requires a series of pairwise coprime integers satisfying some stringent conditions. Therefore, we focus on Shamir secret sharing (SSS) due to the simplicity in generating shares.

A major issue of SSS scheme is that it is inefficient for storage or processing of voluminous data, as there is a data expansion in the process. Krawczyk [26] introduced ramp scheme with shorter share size by relaxing the information theoretic secrecy conditions. In this scheme, the secret to be shared is first encrypted using a length preserving encryption scheme such as AES. Then, this encrypted secret is divided into n fragments using Rabin′s information dispersal algorithm. Then the n shares of encryption key are generated using a perfect secret sharing scheme. The share given to each of the n participants includes the encrypted fragment and its key share. The security of this ramp scheme relies on computational secrecy which means that no computational information can be gained by resource bounded adversaries. The security of this scheme completely depends on the security of encryption and no security is embedded in the information dispersal algorithm. As the key shares are also given to the participants, this scheme cannot be used for privacy preserving cloud computing applications since there is chance of collusion by threshold number of cloud servers. Moreover, since the secret is encrypted with AES before share generation, this scheme cannot be directly used in applications requiring homomorphic processing. This is due to the fact AES encryption spoils the homomorphic property of information dispersal algorithm.

The homomorphic property of SSS [18] is widely used for secure multi party computation (MPC) [27] and various privacy preserving computing applications. A secure MPC framework, Sharemind which utilizes SSS scheme to split the data among three participants is proposed in [28]. Many privacy preserving computing applications such as data mining [29], financial data analysis [30] are proposed based on the protocols for various operations available in Sharemind framework. Another computing library for secure MPC based on SSS scheme, known as SEPIA is proposed in [31] for privacy preserving data aggregation for network security. Algorithms for executing different privacy-preserving SQL queries over the data outsourced using SSS scheme is proposed in [32]. However, all these works are based on the assumption that the number of collaborating participants will be less than the threshold.

Many privacy preserving data processing applications over cloud based on secret sharing schemes have also been introduced in recent years. SSS based image denoising over cloud is proposed in [33], but it can support only integer subtraction and addition operations. A pre-processing method is introduced in [34] through which real number addition/scalar multiplication can be done in shared domain (SD). However, this work allows multiplication/division operation with terminating decimals only. A solution to support arithmetic division operations for nonterminating decimals on the image shares is provided in [35]. All these works [33–35] are aimed at low-level image enhancement tasks such as denoising, anti-aliasing and contrast enhancement in SD over cloud, which involves a 3×3 mean filtering operation. A secure data deduplication scheme over cloud which relies on CRT based secret sharing with is proposed in [36]. This scheme makes use of authentication codes to verify the integrity of data prior to downloading the outsourced data. A reversible data hiding scheme [37] is proposed in which the secret data to claim the ownership is embedded into the image shares generated using CRT based scheme before outsourcing to cloud. An image tampering detection and recovery scheme which relies on secret sharing based on permutation ordered binary (POB) number system is proposed in [38]. In this scheme, the detection and recovery watermarks are embedded into the shares before outsourcing. The problem with all these works based on secret sharing is that the security relies on the assumption that threshold number of participants will not collude.

1.2. Motivation and contributions

Based on the literature mentioned in the related works section, the different cloud computing applications which employ shared domain processing are secure data aggregation, querying, data mining, multimedia processing etc which involve only linear operations on the data. As SSS possess additive homomorphism, it can be adopted in any application which demand linear operations in shared domain. The main drawbacks of SSS scheme are (1) huge storage overhead due to the need for storage of n shares in n CSPs; and (2) vulnerability to collusion attack. The security of the privacy preserving processing methods based on SSS assumes that more than (t - 1) CSPs will not collude. However, SSS cannot guarantee information-theoretic security if at least t CSPs collude. This is due to the fact that when we extend the application of secret sharing schemes for distributed storage, it is highly probable that the third party servers (CSPs) holding the shares collude. Then the (t, n) secret sharing scheme fails to ensure data secrecy if t servers holding shares collude. Therefore in this scenario, we need to ensure that only the authorized entities will be able to reconstruct the secret data by combining t shares stored in CSPs whereas the third party CSPs will not be able to retrieve the secret data through collusion. Hence, we modified SSS scheme to tackle the issues of both storage overhead and collusion attack in such a way that only two shares need to be outsourced to provide adequate security.

But the adoption of the modified SSS scheme to outsource volumetric data such as multimedia data or medical images will be inefficient in terms of storage and communication since the size of each share in the modified SSS scheme is same as that of secret. Therefore, we also propose a modified ramp secret sharing (MRSS) scheme resistant to collusion attack in order to resolve the storage overhead issue while outsourcing volumetric data. Even though the proposed schemes can be used for any secure domain data processing application which involve linear operations, in this paper DWT computation for image data is chosen as the typical application to validate the efficiency of the proposed schemes in privacy preserving data processing due to the wide applicability of DWT in feature extraction, fusion etc. for disease diagnosis from pathological images. The major contributions of this paper are as follows.

1. A modified Shamir secret sharing (MSSS) scheme is proposed to resist collusion attacks. The proposed MSSS scheme provides security against collusion attacks through encrypting the secret before share generation and hiding the index used to generate the shares from the CSP.
   (a) Hiding the index prevents the CSPs from reconstructing the secret using shares during collusion.
   (b) Encrypting the secret before share generation prevents the adversary from mounting the known-plaintext attack during collusion.
   (c) The encryption is done by blinding the secret with random numbers generated from keystream of linear feedback shift register (LFSR) in order to preserve the homomorphic properties of the secret sharing scheme. Different random numbers needed to blind secret are generated through a design based on LFSRs.
2. Even though only two shares are needed in the proposed MSSS scheme, data availability and data integrity can be provided by extending it to a (t, n) scheme. The (t, n) MSSS scheme ensures data availability as long as any t out of n CSPs are available. It can also ensure data integrity and detect tampering of data together with identity of the tampered server if the number of servers corrupted is limited to $n - t - 1$.
3. A modified ramp secret sharing scheme (MRSS) scheme is also proposed to reduce the storage overhead further without sacrificing the data processing capability and security.
4. Privacy preserving DWT computation on images using Haar wavelet is designed based on proposed schemes.
5. The security of the proposed schemes is analyzed through mathematical cryptanalysis and ability to resist different statistical attacks.
6. The performance evaluation of the proposed shared domain (SD) DWT computation is done by qualitative and quantitative analysis of the SD results in comparison with those in plaintext domain (PD).

The rest of the paper is organized as follows. In the next section, we describe the system model and adversary model. Section 3 and Section 4 present the proposed MSSS and MRSS schemes respectively. The integrity verification of reconstructed data based on proposed schemes is detailed in Section 5 and the steps in privacy preserving image decomposition is explained in Section 6. The security analysis is discussed in Section 7 and the performance evaluation of the SD image decomposition and comparison with related schemes is detailed in Section 8, followed by concluding remarks in Section 9.

2. System model and adversary model

We now describe the system model and adversary model for privacy preserving image processing over cloud based on SSS as the SD operation considered is image decomposition.

2.1. System model

The system model for the secure domain storage and image processing based on (t, n) SSS is shown in Fig. 1, where the obfuscated shares of original images are distributed among n different CSPs, who perform the required image processing operations on their corresponding shares. The authorized entity can reconstruct the final processed image by retrieving processed image shares from any of the t CSPs. The data availability can be ensured if any t out of n CSPs are available. Similarly, integrity of the stored and processed data can be verified as long as any $t + 1$ out of n CSPs are not corrupted.

2.2. Adversary model

This work considers both passive adversaries and active adversaries. Passive adversary can eavesdrop on the data stored in any of the CSPs or cloud servers. Whereas, it is assumed that the active adversaries can corrupt data only in atmost $n - t - 1$ out of n cloud servers since shares from at least $t + 1$ servers are required for integrity verification. The cloud servers are considered as semi trusted which implies that they execute the storage and processing tasks properly. However, they may be curious to know the stored data. The adversary can also try to mount known-plaintext attack (KPA) as he may possess knowledge of some plaintext based on the nature of stored data. Chosen plaintext attack (CPA) is not considered as a valid attack in this scenario, as it is impossible for the adversary to have access to the client encrypting machine which may be geographically separated from the cloud storage servers.

3. Proposed modified Shamir secret sharing scheme

In original SSS, the shares corresponding to a secret are generated through polynomial evaluation and the secret can be reconstructed from threshold number of shares through Lagrange interpolation. Here the secret, polynomial coefficients and the indices used to generate shares are elements of finite field, F_p. The details of share generation, secret reconstruction and homomorphic properties are as follows. Table 1 list the primary notations used in our scheme.

3.1. Original Shamir secret sharing scheme

3.1.1. Share generation

In (t, n) SSS, the n shares corresponding to the secret, s_1 (data to be shared) are generated through evaluating a polynomial of degree $t - 1$ as shown in Eq. (1).

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} (mod\ p) \tag{1}$$

The share of the jth participant, $f(x_j)$, $1 \le j \le n$ corresponding to secret, s_1 is obtained by evaluating the polynomial, $f(x)$ with constant term, $a_0 = s_1$ and index, $x_j \in F_p$ as shown in Eq. (2).

$$f(x_j) = s_1 + a_1 x_j + a_2 x_j^2 + \cdots + a_{t-1} x_j^{t-1} (mod\ p) \tag{2}$$

The index x_j is chosen in such a way that $x_n > x_{n-1} > \cdots > x_2 > x_1$. The coefficients a_k, $1 \le k \le t - 1$ of the polynomial are randomly chosen from a uniform distribution over the integers modulo p, where p is prime.

3.1.2. Secret reconstruction

The secret shared among the different participants can be reconstructed from any t out of n shares through Lagrange interpolation. The idea is to construct a set of t polynomials known as Lagrange base polynomials, $\gamma_j(x)$, $1 \le j \le t$ such that

$$\gamma_j(x_i) = \begin{cases} 0, i = j \\ 1, i \neq j \end{cases} \tag{3}$$

These Lagrange base polynomials, $\gamma_j(x)$ can be easily constructed by placing the roots appropriately and then normalizing the result such that $\gamma_j(x_j) = 1$. Thus, the expression for $\gamma_j(x)$ can be written as,

$$\gamma_j(x) = \prod_{k=1, k\neq j}^{t} \frac{(x - x_k)}{(x_j - x_k)} \tag{4}$$

Then the final interpolating polynomial, $f(x)$ can be expressed as the weighted sum of the Lagrange polynomials, $\gamma_j(x)$, where share values, $f(x_j)$ form the corresponding weights as shown in Eq. (5).

$$f(x) = \sum_{j=1}^{t} f(x_j).\gamma_j(x) \tag{5}$$

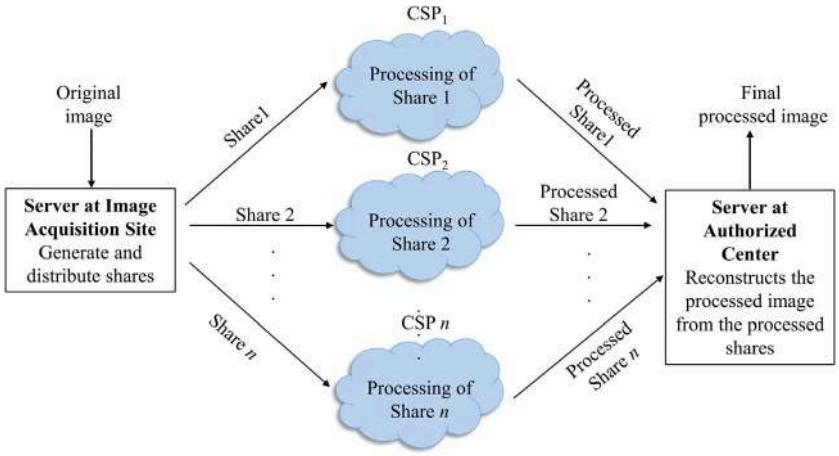

Fig. 1. Proposed Scheme for shared domain storage and image processing.

Table 1
Notations in our scheme.

Notation	Description
F_p	Finite field of size p
t	Threshold no. of shares in (t, n) SSS
n	Total no. of shares in (t, n) SSS
s_i	ith secret data in MSSS scheme
$f(x)$	Polynomial used for share generation
x_j	Indices used to generate shares by evaluating $f(x)$
γ_j	Lagrange coefficient of jth share needed to reconstruct secret
y_{ij}	jth share of secret s_i
L	Length of LFSR
$q(z)$	Feedback polynomial of LFSR
k_1	Initial state or secret key of LFSR1
β_1	Initial state of LFSR2 used to derive random multiplier
r_{ij}	jth random number from ith linearly independent LFSR keystream
s_{ij}	jth secret data in the ith secret vector in MRSS scheme
$\bar{s}_{ij}$	Blinded secret data corresponding to s_{ij}
$\bar{s}_r$	Blinded reconstructed data vector
k'	No. of corrupted servers
N_c	No. of possible combinations of t servers generating corrupted data
Z'_i	List of indices of t servers with $\binom{n}{t}$ entries
Z_i	List of indices of t servers from Z'_i which gives same $\bar{s}_r$
i_c	Indices of corrupted servers

In original SSS scheme, as the secret, s_1 is the constant term of the polynomial, it can be reconstructed by evaluating the Eq. (5) with $x = 0$. Thus, the secret, $s_1 = f(0)$ reconstructed from any t out of n shares through Lagrange interpolation is as shown in Eq. (6).

$$\begin{aligned} s_1 &= \sum_{j=1}^{t} f(x_j) \cdot \gamma_j(0) \\ &= \sum_{j=1}^{t} f(x_j) \cdot \prod_{k=1,k\neq j}^{t} \frac{x_k}{(x_k - x_j)} \ (mod\ p) \end{aligned} \tag{6}$$

3.1.3. Homomorphic property

SSS supports additivity and homogeneity properties which are the requirements of an additive HES. Let $s_1, s_2 \in F_p$ be two secrets to be shared using SSS and $[s_1]_j^t$, $[s_2]_j^t$ be the corresponding shares. Then the homomorphic properties can be defined as,

$$\begin{aligned} [s_1]_j^t + [s_2]_j^t &= \left[s_1 + \sum_{k=1}^{t-1} a_k * x_j^k \ (mod\ p) \right] \\ &+ \left[s_2 + \sum_{k=1}^{t-1} b_k * x_j^k \ (mod\ p) \right] \\ &= \left[(s_1 + s_2) + \sum_{k=1}^{t-1} c_k * x_j^k \ (mod\ p) \right] \\ &= [s_1 + s_2]_j^t \end{aligned} \tag{7}$$

$$\begin{aligned} \alpha \cdot [s_1]_j^t &= \alpha \cdot \left[s_1 + \sum_{k=1}^{t-1} a_k * x_j^k \ (mod\ p) \right] \\ &= \left[\alpha s_1 + \sum_{k=1}^{t-1} d_k * x_j^k \ (mod\ p) \right] \\ &= [\alpha s_1]_j^t \end{aligned} \tag{8}$$

where $\alpha \in F_p$ is a scalar. Here $c_k = a_k + b_k$ and $d_k = \alpha a_k$ represent two different random coefficients in F_p resulting from the addition and scalar multiplication of random coefficients respectively. Eqs. (7) and (8) indicate that the reconstruction of the linear combination of the secret shares yields the same value as that obtained by computing the linear combination on the original secrets. This is due to the fact that the reconstruction of the secret shares does not depend on the random coefficients and it depends only on the indices used to generate the shares as shown in the Lagrange interpolation formula (Eq. (6)).

In original (t, n) SSS, both the share $f(x_j)$ and the associated index, x_j are sent to the participant (CSP). SSS provides information theoretic security since each share of the secret is stored in different CSP. However, if any of the t CSPs collude, the secret will be revealed. Hence, the major problem with the SSS is its susceptibility to collusion attack. In this paper, we have modified the SSS scheme as given in the following section so as to resist collusion attack.

3.2. Design of modified Shamir secret sharing scheme

In original (t, n) SSS scheme, the data owner generates the n shares, $f(x_j)$, $1 \le j \le n$ of secret data using indices x_j. Then the n shares, $f(x_j)$, $1 \le j \le n$ and its corresponding index, x_j are outsourced to n different cloud service providers (CSPs) so that the secret data can be reconstructed by retrieving shares from any of the t CSPs. However, the secret shared among different CSPs using original (t, n) SSS scheme can be retrieved through collusion of t CSPs using Lagrange interpolation formula given in Eq. (6). To perform this, each CSP, say j should know its share, $f(x_j)$ and the corresponding index, x_j. Hence in order to prevent this collusion attack, we propose to hide the index from the CSPs. The secret indices are distributed only among the data owners through key exchange schemes or by encrypting the keys and communicating them prior to data sharing. Therefore the index, x_j acts as key for the modified SSS. This will not affect the homomorphic processing since the homomorphic operations are carried out only on the shares. The indices are required only during the reconstruction of secret data which is performed by the data owner or any other authorized entity by retrieving processed data shares from any of the t

CSPs as shown in Fig. 1. However, the adversary can mount a known-plaintext attack (KPA) during collusion mentioned in [39] even if the index is kept secret. The details of this KPA through collusion [39] if any t secrets and their shares are known to the adversary is described below.

Let $y_{ij} = f(x_j)$ represents the jth share of secret, s_i. If the adversary knows t secrets, $s_i, 1 \le i \le t$ and their corresponding shares, $y_{ij}, 1 \le j \le t$ then he can find the Lagrange base polynomial, $\gamma_j (= \gamma_j(0))$ by solving the following set of equations [39].

$$\begin{aligned} y_{11}\gamma_1 + y_{12}\gamma_2 + \cdots + y_{1t}\gamma_t - s_1 &\equiv 0 \ (mod\ p) \\ y_{21}\gamma_1 + y_{22}\gamma_2 + \cdots + y_{2t}\gamma_t - s_2 &\equiv 0 \ (mod\ p) \\ &\vdots \\ y_{t1}\gamma_1 + y_{t2}\gamma_2 + \cdots + y_{tt}\gamma_t - s_t &\equiv 0 \ (mod\ p) \end{aligned} \tag{9}$$

Once the adversary obtains the Lagrange bases $\gamma_1, \gamma_2, \ldots, \gamma_t$, he can use this to retrieve other secrets shared among the colluding CSPs. To prevent this collusion and to further enhance the security of SSS, we propose the following modifications to SSS.

In order to prevent the adversary from retrieving the Lagrange bases $\gamma_1, \gamma_2, \ldots, \gamma_t$ by solving the above set of equations, it is required to encrypt the secret before share generation. However, encrypting the secret using complex encryption scheme eliminates the inherent additive homomorphism provided by the SSS as traditional encryption schemes cannot support encrypted domain processing. Hence, we propose to encrypt the secret by adding it to random numbers generated through a key. Thus, after blinding the secrets, the above set of equations can be rewritten as,

$$\begin{aligned} y_{11}\gamma_1 + y_{12}\gamma_2 + \cdots + y_{1t}\gamma_t - (s_1 + r_1) &\equiv 0 \ (mod\ p) \\ y_{21}\gamma_1 + y_{22}\gamma_2 + \cdots + y_{2t}\gamma_t - (s_2 + r_2) &\equiv 0 \ (mod\ p) \\ &\vdots \\ y_{t1}\gamma_1 + y_{t2}\gamma_2 + \cdots + y_{tt}\gamma_t - (s_t + r_t) &\equiv 0 \ (mod\ p) \end{aligned} \tag{10}$$

where $r_1, r_2, \ldots, r_t$ are the random numbers added to the t secrets. Blinding the secret can prevent t colluding CSPs from retrieving γ_j by solving the set of t equations, since it consists of $2t$ unknowns as given in Eq. (10).

Now, in order to ensure that the secrecy offered by random numbers is not spoiled during processing operation on shares, the method for generation of random numbers need to be developed through a careful design. The exact design methodology depends upon the data processing operation required to be performed in SD. Following section discusses the design requirements to be considered in random number generation for linear operations in SD.

3.3. Design of random numbers

The random number generator needs to be designed carefully to ensure that the randomness and security properties of random numbers are preserved in the homomorphically combined shares. Moreover, it is desirable to develop the method for generation of random numbers with very low complexity of operations as blinding needs to be performed at the client side. Linear feedback shift register (LFSR) based keystreams are good candidates due to their good statistical properties and low structural complexity [40]. Further, the LFSR keystream possess homomorphic properties required to provide additive homomorphism during SD processing. This implies that linear combinations of keystreams should yield another keystream with the same randomness properties, which is established through Theorem 1. The secret key of the LFSR is formed by the initial state, $k(z)$ and feedback polynomial, $q(z)$ that determines the feedback connections.

Theorem 1. *The LFSR keystreams satisfy additivity and homogeneity properties.*

Additivity property –The addition of the LFSR keystreams will produce a new keystream, generated from an initial state which corresponds to the sum of initial states of individual keystreams.

Homogeneity property –The scalar multiplication of an LFSR keystream will produce a new keystream, generated from an initial state which corresponds to the scalar multiple of the initial state of original keystream.

Proof. See Appendix A □

But due to the linearity property of LFSR keystream, the linear combination during SD computation may result in null keystream. This can be prevented through employing symbols from linearly independent keystreams for blinding the secrets to be linearly combined. From Theorem 1, it is clear that the linearly independent LFSR initial states generate linearly independent keystreams. But an LFSR of length L can generate only L linearly independent initial states since every state of this LFSR is an L-dimensional vector in the vector space W over F_p. We also propose a method for deriving the L linearly independent initial states from an initial secret key, $k = k_0, k_1, \ldots, k_{L-1}$ through cyclic shifting operation. The conditions required to generate L linearly independent keystreams from initial key, k is established through Theorem 2.

Theorem 2. *The set of L initial states of an LFSR of length L derived through cyclic shift of the initial secret key $k(z)$ will be linearly independent, if gcd $(k(z), z^L - 1)$ forms a polynomial of zero degree.*

Proof. See Appendix B □

If LFSR initial states used for generating linearly independent keystreams are computed from the initial secret key through simple linear shifting operations, it may lead to security leakage. This is due to the fact that all the keystreams used for blinding the image will be simply the shifted versions of one keystream. Therefore, to strengthen the security, the initial state required for producing the linearly independent keystreams are generated through the cyclic shift of previous initial states followed by multiplication by a random number from F_p. Thus, security leakage can be prevented while retaining randomness properties and linear independence.

Algorithm 1 details the steps for generating linearly independent keystreams required to obtain the random numbers used to blind the secrets, whose shares are to be processed in the cloud. M linearly independent keystreams, $r_{ij}, 1 \le i \le M, 1 \le j \le (p^L - 1)$ are generated as the output of this algorithm, where M is the number of secrets that are linearly combined during SD computation. Two LFSRs are used for generating the required keystreams, in which LFSR1 generates the keystream used for creating random numbers whereas LFSR2 derives the random multiplier needed for updating the states of LFSR1. The inputs to the algorithm are initial state or key of LFSR1, k_1; feedback polynomial, $q(z)$; and initial key, β_1 of LFSR2 needed to derive random multiplier β_i. Here LFSR1-PRNG refers to the LFSR based pseudo random number generator which generates M keystreams from the initial state, k_1 and feedback polynomial, $q(z)$. LFSR2-STATE UPDATE indicates the updation of initial state, β_1 of LFSR2, which generates a single random symbol as output. In the algorithm, $D_{Ri}(k_1)$ denotes that the initial state, k_1 is right shifted by i bits.

3.3.1. Keyspace

The secret keys of the proposed MSSS scheme consists of initial state of LFSR1, k_1; feedback polynomial, $q(z)$; initial key of multiplier, β_1 and n indices, x_j, $1 \le j \le n$ used to generate n shares.

Algorithm 1 For generating linearly independent keystreams

Input: k_1, β_1, $q(z)$
Output: r_{ij}
1: **for** $i = 1 : M$ **do**
2: r_{ij} = LFSR1-PRNG(k_i)
3: $k_{i+1} = D_{Ri}(k_1) \cdot \beta_i$
4: β_{i+1} = LFSR2-STATE UPDATE(β_i)
5: **end for**
6: **return** r_{ij}

Table 2
Comparison of storage overhead between RS codes, SSS and RSS scheme based DSS for $(t, n) = (3, 4)$

Data storage technique	Storage overhead
RS Code	1.33 × data size
SSS Scheme	4 × data size
RSS Scheme	1.33 × data size

4. Proposed modified ramp secret sharing for reduced storage overhead

Cloud platform is usually built upon distributed storage system (DSS), which relies on either replication codes or erasure codes such as Reed Solomon (RS) codes for ensuring data availability. The same data is replicated among n servers using $(1, n)$ replication codes which imposes a storage overhead of n times. Whereas a fraction $(1/t)$ of the data is stored in each of the n servers using (t, n) RS codes which reduces the storage overhead to n/t times. SSS is a special case of RS codes, where only one coefficient is used to store the secret instead of the entire polynomial. This implies that in RS codes, all the t coefficients in Eq. (1) are filled with the data to be stored whereas in SSS, the secret to be shared is assigned as the constant term a_0 and all other $t-1$ coefficients are filled with random values. Thus, the storage overhead for SSS will be n times, i.e., similar to that of $(1, n)$ replication codes [5]. Hence while extending proposed MSSS to provide data availability in addition to SD processing over cloud, the storage overhead also increases. However, the storage overhead problem of MSSS can be resolved by using a variant of secret sharing scheme, i.e., ramp secret sharing (RSS) scheme [41], which fills all the coefficients in the polynomial with the data to be shared, instead of original SSS. Hence, (t, n) RSS scheme is similar to RS codes and ensure data availability with lesser storage overhead compared to SSS. Table 2 shows the comparison of storage overhead of various schemes for $t = 3$ and $n = 4$.

In addition to reduction in storage overhead, RSS can also support SD processing. However, original RSS scheme also suffers from collusion attack. Hence, it is required to modify RSS scheme to resist collusion attack without sacrificing the data processing capability. The details of the modified RSS scheme are as follows.

4.1. Design of modified ramp secret sharing scheme

In original (t, n) RSS, the n shares are generated through polynomial evaluation as in SSS, but all the t random coefficients in the polynomial are replaced with the secret data to be shared as shown in Eq. (11), where s_{11}, s_{12}, ⋯, s_{1t} denote the secret data.

$$f(x) = s_{11} + s_{12}x + s_{13}x^2 + \cdots + s_{1t}x^{t-1} (mod\ p) \tag{11}$$

The share of the jth participant among n participants is obtained by evaluating the polynomial, $f(x)$ given in Eq. (11) with index, $x_j \in F_p$. The secret, s_{11}, s_{12}, ⋯, s_{1t} can be reconstructed by retrieving any t out of n shares.

As in the case of MSSS scheme, the index is hidden from the participant (CSP) through encryption. However, the secrets can be revealed through mounting a KPA during collusion of any t CSPs. In order to reconstruct the secret from any t out of n shares in RSS, it is required to find the Lagrange base polynomial, $\gamma_j(x)$ in terms of its coefficients, $\gamma_{j,i}$ as shown in Eq. (12).

$$\gamma_j(x) = \prod_{k=1,k\neq j}^{t} \frac{(x - x_k)}{(x_j - x_k)} = \sum_{i=1}^{t} \gamma_{j,i} x^i \tag{12}$$

Now, the secret, $s_{11}, s_{12}, \ldots, s_{1t}$ in RSS scheme can be retrieved using coefficients, $\gamma_{j,i}$ as,

$$\begin{bmatrix} s_{11} \\ s_{12} \\ \vdots \\ s_{1t} \end{bmatrix} = \begin{bmatrix} \gamma_{1,1} & \gamma_{2,1} & \cdots & \gamma_{t,1} \\ \gamma_{1,2} & \gamma_{2,2} & \cdots & \gamma_{t,2} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{1,t} & \gamma_{2,t} & \cdots & \gamma_{t,t} \end{bmatrix} . \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_t) \end{bmatrix} \tag{13}$$

This means that the secret reconstruction process can be expressed as a set of t linear equations in terms of the Lagrange coefficients, $\gamma_{j,i}$ and the corresponding shares. Therefore an adversary knowing t different set of secrets can find the Lagrange coefficients by solving these set of linear equations similar to the set of equations shown in Eq. (9) for SSS scheme.

As in the case of MSSS scheme, blinding the secrets can prevent t colluding CSPs from retrieving Lagrange coefficients, $\gamma_{j,i}$ by solving the set of t equations. Hence, the share generation step for share j corresponding to secret, $(s_{11}, s_{12}, s_{13}, \ldots, s_{1t})$ in the case of modified RSS (MRSS) scheme can be written as,

$$f(x_j) = (s_{11} + r_{11}) + (s_{12} + r_{12})x_j + (s_{13} + r_{13})x_j^2 + \cdots + (s_{1t} + r_{1t})x_j^{t-1} (mod\ p) \tag{14}$$

where $r_{11}, r_{12}, r_{13}, \ldots, r_{1t}$ are random numbers used to blind the secret, $s_{11}, s_{12}, s_{13}, \ldots, s_{1t}$ respectively.

MRSS also preserves additivity and homogeneity properties shown in Eq. (7) and (8) for MSSS scheme, which are required to support shared domain processing. In order to ensure that the randomness and security properties of random numbers are preserved in the homomorphically combined shares, the random numbers used to blind secrets in MRSS scheme are also generated using LFSR based keystream generator detailed in Section 3.3. In both (t, n) MSSS and MRSS schemes, the data owner generates n shares of the blinded secret data using indices and only the shares are outsourced to n different cloud service providers (CSPs) so that the secret data can be reconstructed by retrieving shares from any of the t CSPs. In order to prevent the secret data recovery through collusion attack, the indices and LFSR keys are hidden from the CSPs and are distributed only among the data owners through key exchange schemes or by communicating the encrypted keys prior to data sharing, similar to MSSS scheme.

Even though the proposed MRSS scheme looks structurally similar to Krawczyk's scheme [26] detailed in Section 1.1, the indices used to generate shares from the encrypted/blinded secret is hidden from the participants/ cloud servers in the MRSS scheme. Hence, it is difficult to retrieve even the encrypted secret in MRSS scheme if threshold number of cloud servers collude. Moreover, since the security of Krawczyk's scheme depends completely on the security of encryption scheme, it is necessary to use secure encryption schemes such as AES. This prevents the adoption of Krawczyk scheme in cloud based homomorphic processing applications. In the proposed scheme, in order to retain the inherent homomorphic properties of secret sharing scheme, we blind the secret to be processed in SD with carefully designed linearly independent keystreams of LFSR.

5. Proposed scheme for integrity verification and data reconstruction

The integrity of the stored data or processed data can be verified by comparing the data reconstructed from two set of shares. Even though both MSSS and MRSS schemes can facilitate data integrity verification (IV), the IV process is explained with respect to MRSS scheme as it is the generalized version. Consider a (t, n) MRSS scheme, which can tolerate corruption of data stored in any of the $n-t-1$ servers. Here the data file to be stored is first divided into t fragments and it forms the secret data set, say $s_{11}, s_{12}, \ldots, s_{1t}$. The shares, $f(x_j), 1 \le j \le n$ corresponding to this secret data set are generated as follows, where $\tilde{s}_{1j} = s_{1j} + r_{1j}$.

$$f(x_j) = \tilde{s}_{11} + \tilde{s}_{12}x_j + \cdots + \tilde{s}_{1t}x_j^2 (mod\ p) \tag{15}$$

Each of these shares are stored in separate cloud servers. The shares from any $(t+1)$ servers are required for IV. The steps in verifying the integrity of the reconstructed data are detailed in Algorithm 2. During IV, blinded secret, $\tilde{s}'_r$ and $\tilde{s}''_r$ are reconstructed from two set of t servers, $f(x_1), f(x_2), \ldots, f(x_t)$ and $f(x_2), f(x_3), \ldots, f(x_{t+1})$ respectively. The decision on the integrity of the reconstructed data is done by comparing $\tilde{s}'_r$ with $\tilde{s}''_r$. The reconstructed data is correct if they are same, otherwise data is corrupted and measures for data recovery and identification of corrupted servers need to be initiated.

Algorithm 2 For integrity verification of reconstructed data

Input: $f(x_1), f(x_2), \cdots, f(x_{t+1})$: Shares from $t+1$ servers
Output: Status of integrity verification

1: Compute blinded secret, $\tilde{s}'_r = (\tilde{s}'_{11}, \tilde{s}'_{12}, \cdots, \tilde{s}'_{1t})$ using Eq. (13) from shares, $f(x_1), f(x_2), \cdots, f(x_t)$.
2: Compute blinded secret, $\tilde{s}''_r = (\tilde{s}''_{11}, \tilde{s}''_{12}, \cdots, \tilde{s}''_{1t})$ using Eq. (13) from shares, $f(x_2), f(x_3), \cdots, f(x_{t+1})$.
3: **if** $\tilde{s}'_r = \tilde{s}''_r$ **then**
4: **Return** "Reconstructed data is correct".
5: **else**
6: **Return** "Data is corrupted and recovery needed".
7: **end if**

The recovery of secret data and identification of corrupted servers are possible if atmost $n-t-1$ servers are corrupted by adversaries as shares from $t+1$ servers are required for IV. Once the data corruption is detected, in order to identify the malicious server, the shares from all the servers are retrieved and data is reconstructed using all possible combinations of t servers. In (t, n) MRSS scheme, secret set of size t can be reconstructed from t out of n shares in $\binom{n}{t}$ possible ways. The reconstructed data will be incorrect if one or more of the t servers selected for secret reconstruction is corrupted. If one server among n servers is malicious, the number of possible combinations in which the selected set of t servers includes the malicious server can be expressed as $\binom{n-1}{t-1}$. This is equivalent to choosing $t-1$ from $n-1$ servers as the corrupted server is fixed as one of the chosen servers. Similarly, if 2 out of n servers are corrupted, the number of possible combinations in which reconstructed data will be incorrect is given by $\left[\binom{2}{1}\binom{n-1}{t-1} - \binom{2}{2}\binom{n-2}{t-2}\right]$, where $\binom{2}{1}\binom{n-1}{t-1}$ indicates the number of ways in which each of the two servers are in the selected t servers and $\binom{2}{2}\binom{n-2}{t-2}$ denotes the number of ways in which both the corrupted servers are in the selected set of t servers. In general, if k' out of n servers are corrupted, the number of possible combinations of t servers, N_c which generate corrupted secret data is given by Eq. (16). It is to be noted that to ensure recovery of correct data, $k' \le (n-t-1)$.

$$N_c = \binom{k'}{1}\binom{n-1}{t-1} - \binom{k'}{2}\binom{n-2}{t-2} - \binom{k'}{3}\binom{n-3}{t-3} - \cdots - \binom{k'}{k'}\binom{n-k'}{t-k'} \tag{16}$$

Based on the above discussions, the steps in the recovery of secret data and identification of malicious server are developed as detailed in Algorithm 3. As a first step, blinded secret, $\tilde{s}'_{ri} = (\tilde{s}'_{11}, \tilde{s}'_{12}, \ldots, \tilde{s}'_{1t})$, $1 \le i \le \binom{n}{t}$ is computed from all possible $\binom{n}{t}$ combinations of t servers, $Z'_i = (i_1, i_2, \ldots, i_t)$, where each i_k, $1 \le k \le t$ represent the index j, $1 \le j \le n$ of servers. Then a table of $\binom{n}{t}$ entries is created which consists of the indices of t servers, Z'_i and the corresponding computed data, $\tilde{s}'_{ri}$, $1 \le i \le \binom{n}{t}$. The data recovery and detection of the corrupted server can be done by comparing the table entries. The maximum number of table entries which will result in erroneous recovery of data is decided by the value of N_c computed with $k' = (n-t-1)$, as it is possible to recover the data without tampering only if $k' \le (n-t-1)$. This means that the data can be recovered only if the number of table entries, N_m with matching $\tilde{s}'_{ri}$ is at least $\left(\binom{n}{t} - N_c\right)$. Then the corresponding matching data, $\tilde{s}'_{ri}$ in N_m entries, represented as $\tilde{s}_r = (\tilde{s}_{11}, \tilde{s}_{12}, \ldots, \tilde{s}_{1t})$ give the recovered blinded secret and the corrupted servers can be identified from the list of indices of servers which give erroneous reconstructed data. It must be noted that the indices of malicious servers, i_c, will not be present in the list of indices of servers, Z_i which return the correct reconstructed data. On the other hand, if the retrieved data values, $\tilde{s}'_{ri} = (\tilde{s}'_{11}, \tilde{s}'_{12}, \ldots, \tilde{s}'_{1t})$ do not match in at least $\left(\binom{n}{t} - N_c\right)$ table entries, it indicates that the data recovery is not possible as the number of corrupted servers, $k' > (n-t-1)$.

Algorithm 3 For data recovery and corrupted server identification

Input: $f(x_1), f(x_2), \cdots, f(x_n)$: Shares from n servers
Output: Recovered blinded secret, $\tilde{s}_r = (\tilde{s}_{11}, \tilde{s}_{12}, \cdots, \tilde{s}_{1t})$ and indices of corrupted servers, i_c

1: Assign unique index j, $1 \le j \le n$ for each server.
2: **for** $i = 1 : \binom{n}{t}$ **do**
3: Pick ith set of t servers with indices $(i_1, i_2, \cdots, i_t)$ where i_k, $1 \le k \le t$ represent the index j of servers.
4: Compute blinded secret, $\tilde{s}'_{ri} = (\tilde{s}'_{11}, \tilde{s}'_{12}, \cdots, \tilde{s}'_{1t})$ from shares $f(x_{i_1}), f(x_{i_2}), \cdots, f(x_{i_t})$.
5: **end for**
6: Create a table of computed data $\tilde{s}'_{ri}$, $1 \le i \le \binom{n}{t}$ together with set of indices, $Z'_i = (i_1, i_2, \cdots, i_t)$, where i_k is the index of servers participated in reconstructing the data.
7: Compute N_c for $k' = (n-t-1)$.
8: Compute the number of entries, N_m with matching $\tilde{s}'_{ri}$ and represent those $\tilde{s}'_{ri}$ as $\tilde{s}_r$ and corresponding Z'_i as Z_i.
9: **if** $N_m \ge \left(\binom{n}{t} - N_c\right)$ **then**
10: Find the indices of corrupted servers, i_c, where $i_c \notin Z_i$ which return the correct reconstructed data, $\tilde{s}_r$.
11: **Return** $\tilde{s}_r = (\tilde{s}_{11}, \tilde{s}_{12}, \cdots, \tilde{s}_{1t})$
12: **Return** i_c
13: **else**
14: **Return** "More than $(n-t-1)$ servers are corrupted and data cannot be recovered".
15: **end if**

Table 3 shows the identification of corrupted servers by taking (3, 6) MRSS scheme as an example. This scheme can tolerate data corruption in any of the 2 servers as $n-t-1 = 2$. Therefore, for the results tabulated in Table 3, malicious server identification with one corrupted server (say, server, $i_c = 3$) and two corrupted servers (say, server, $i_c = 3$ and 5) are considered as two cases. The number of combinations, N_c through which the selected set of t servers include one corrupted server and two corrupted servers respectively are 10 and 16. These values can be obtained by substituting $k' = 1$ and $k' = 2$ in Eq. (16). In Table 3, the tick mark entries show that the data reconstructed from servers with the corresponding set of indices are same. Whereas cross mark indicate that different sets of data are obtained on reconstruction using the corresponding set of servers.

In order to relieve the data owner from the burden of verifying the integrity of reconstructed data, it is possible to assign the IV task to a trusted third party (TTP), who will not collude with the cloud servers.

Table 3
Corrupted server identification.

Sl No. i	Indices of t servers $Z'_i = (i_1, i_2, i_3)$	One corrupted server ($i_c = 3$)	Two corrupted servers ($i_c = 3$ & 5)	Sl No i	Indices of t servers $Z'_i = (i_1, i_2, i_3)$	One corrupted server ($i_c = 3$)	Two corrupted servers ($i_c = 3$ & 5)
1	1 2 3	X	X	11	2 3 4	X	X
2	1 2 4	✓	✓	12	2 3 5	X	X
3	1 2 5	✓	X	13	2 3 6	X	X
4	1 2 6	✓	✓	14	2 4 5	✓	X
5	1 3 4	X	X	15	2 4 6	✓	✓
6	1 3 5	X	X	16	2 5 6	✓	X
7	1 3 6	X	X	17	3 4 5	X	X
8	1 4 5	✓	X	18	3 4 6	X	X
9	1 4 6	✓	✓	19	3 5 6	X	X
10	1 5 6	✓	X	20	4 5 6	✓	X

However, the data owner should ensure that TTP should not be able to access the original secret data. This condition is also satisfied here since the TTP will only be able to get the randomized secret data as the blinded secrets are generated during IV using share indices.

6. Privacy preserving DWT computation based on proposed MSSS and MRSS scheme

The exact design methodology of random numbers depends on the SD data processing operations. DWT is one of the primary operations performed on medical images before many other more sophisticated tasks such as anomaly detection, fusion etc. The shares are generated corresponding to each pixel of the image in MSSS whereas the shares are generated corresponding to a group of pixels (group size depends on t value) in MRSS. For better understanding, first the SD DWT computation is explained with respect to (t, n) MSSS scheme. This is followed by the modifications to be done while converting it to (t, n) MRSS scheme. In order to illustrate the design of random numbers, image decomposition based on single level Haar wavelet is considered. The privacy preserving image decomposition consist of mainly three steps: namely share generation, Haar DWT computation in SD and reconstruction of decomposed image. The detailed explanation of each of the three steps are given in the following section.

6.1. Share generation

The shares are generated at the client side, where image acquisition take place as shown in Fig. 1. Let I be the original image of size $b \times b$, which is to be outsourced to the cloud servers for SD DWT computation. Share generation process consists of 3 steps: blinding the original image, preprocessing and generation of image shares.

(i) Blinding the original image
Initially the image pixel values are blinded by adding with random values generated through key. The random values taken from LFSR keystream are arranged as a 2D matrix to form the random image of the same size as original image for this purpose.
The blinding process can be mathematically represented as,

$$\tilde{I}(u, v) = I(u, v) + R(u, v) \ (mod\ p) \tag{17}$$

where $1 \leq u, v \leq b$. R represents the random image used for blinding the original image, I. Here the pixel values of images, I and R are symbols from field, F_p.
While decomposing image share using Haar wavelet, the row-wise operations are followed by column-wise operations. The single-level Haar wavelet decomposition is demonstrated in Fig. 2 with a toy example consisting of a 4×4 image. From Fig. 2, it is clear that the linear combination of adjacent 4 pixels are taken during single level Haar DWT computation. Hence the corresponding 4 random values in the random image need to be taken from linearly independent LFSR keystreams, so that security properties are unaffected during single level Haar wavelet decomposition.

(ii) Preprocessing the blinded images
Preprocessing is necessary to assure that the image pixel values after decomposition are integers. Since single level Haar DWT requires an averaging operation which involves a division by 2 (Eq. (24)), the blinded image is preprocessed as,

$$\hat{I}(u, v) = \tilde{I}(u, v) \times 2 \tag{18}$$

(iii) Generation of image shares
Using (t, n) MSSS scheme, the jth share corresponding to the image to be outsourced to the jth CSP, where $1 \leq j \leq n$, is generated from the preprocessed image as,

$$\hat{I}_j(u, v) = \hat{I}(u, v) + \sum_{k=1}^{t-1} a_k(u, v) \times x_j \ (mod\ p) \tag{19}$$

where $\hat{I}_j$ represents the jth share of preprocessed image, $\hat{I}$; and a_k represent the random coefficients used for share generation. x_j denotes the index of jth image share and are random integers from F_p. It is to be noted that in MRSS scheme, the $t-1$ random coefficients a_k will be replaced by pixel values of the image during share generation.

6.2. 2D Haar DWT computation in SD

The CSPs compute the DWT on their image shares. As demonstrated in Fig. 2, the DWT using Haar wavelet is computed by first performing row-wise decomposition, which is then followed by column-wise decomposition. The steps in the single-level Haar DWT computation are as follows.

(i) Row-wise decomposition
The row-wise decomposition of each of the jth share of image is given by the following pair of equations.

$$\hat{I}_{Rj}(u, v) = \hat{I}_j(u, w) + \hat{I}_j(u, w+1) \ (mod\ p) \tag{20}$$

$$\hat{I}_{Rj}(u, (b/2) + v) = \hat{I}_j(u, w) - \hat{I}_j(u, w+1) \ (mod\ p) \tag{21}$$

where $\hat{I}_{Rj}$ denotes the row decomposed version of jth image share, where $1 \leq j \leq n$. Here, for each value of u, v is varied from 1 to $b/2$ and w is varied from 1 to b by incrementing in steps of 2. Thus u and w values range from 1 to b and v values range from 1 to $b/2$.

(ii) Column-wise decomposition
Now, the column wise decomposition is performed on the row decomposed image share according to the pair of equations given below.

$$\hat{I}_{Cj}(u, v) = \hat{I}_{Rj}(w, v) + \hat{I}_{Rj}(w+1, v) \ (mod\ p) \tag{22}$$

$$\hat{I}_{Cj}((b/2) + u, v) = \hat{I}_{Rj}(w, v) - \hat{I}_{Rj}(w+1, v) \ (mod\ p) \tag{23}$$

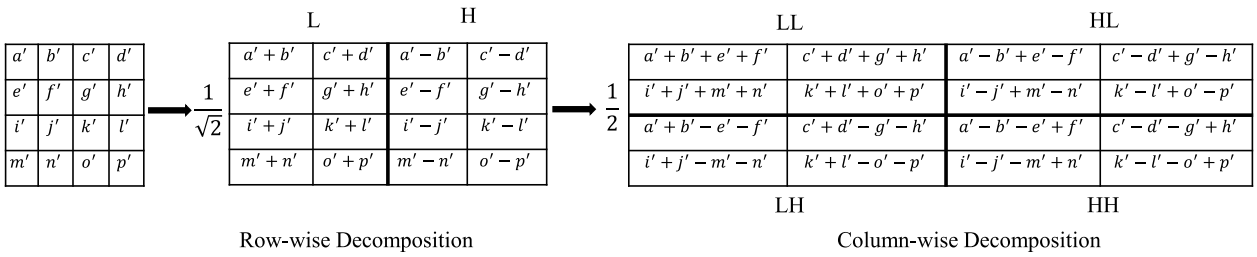

Fig. 2. Single level Haar wavelet decomposition.

where $\hat{I}_{Cj}$ denotes the column decomposed version of row decomposed jth image share. Here also w ranges from 1 to b (in steps of 2), whereas u ranges from 1 to $b/2$ and v ranges from 1 to b.

(iii) Division by two
In order to obtain the final decomposed image, each pixel of column-wise decomposed image is divided by 2 as per the following equation.

$$\hat{I}_{Dj}(u,v) = \hat{I}_{Cj}(u,v) \times (2^{-1} \ (mod\ p)) \ (mod\ p) \tag{24}$$

where $\hat{I}_{Dj}$ represents the final decomposed jth image share.

6.3. Reconstruction of decomposed image

Now in order to obtain original decomposed image, the health care provider or authentic entity should retrieve the decomposed image shares from any of the t CSPs. The steps in the recovery of the final decomposed plaintext image are as follows.

1. Lagrange interpolation
The decomposed image, $\hat{I}_D^R(u,v)$ in blinded form is reconstructed from any t shares using Lagrange interpolation as follows.

$$\begin{aligned}\hat{I}_D^R(u,v) = \ & \hat{I}_{D1}(u,v) * \gamma_1 + \hat{I}_{D2}(u,v) * \gamma_2 + \cdots \\ & + \hat{I}_{Dt}(u,v) * \gamma_t \ (mod\ p)\end{aligned} \tag{25}$$

where

$$\gamma_j = \prod_{1 \le k \le t, k \ne j} \frac{x_k}{x_k - x_j} \tag{26}$$

It is to be noted that in MRSS scheme, the decomposed image in blinded form is reconstructed by multiplying the set of t shares with the Lagrange coefficient matrix as shown in Eq. (13).

2. Post processing
As the final value could be positive or negative, care need be taken to select p such that it is at least twice larger than the range of values involved. Then the final value, $\hat{I}_D^P$ can be correctly recovered in the following way.

$$\begin{aligned}\hat{I}_D^P(u,v) &= \hat{I}_D^R(u,v)/2; \ if \ \hat{I}_D^R(u,v) > (p+1)/2 \\ &= (\hat{I}_D^R(u,v) - p)/2; \ if \ \hat{I}_R^D(u,v) < (p+1)/2\end{aligned} \tag{27}$$

3. Removal of processed random image
As these SD processing operations are done on the blinded image shares, it is required to cancel the effect of random image from the reconstructed decomposed image. The random image also undergoes same processing performed on the original image, during DWT computation. Therefore, the decomposed random image, $\hat{R}_D^P$ can be generated by applying the same linear combination on the random numbers in the random image. Then the final image in DWT domain can be obtained as,

$$\hat{I}_D(u,v) = \hat{I}_D^P(u,v) - \hat{R}_D^P(u,v) \tag{28}$$

Original Image, I

a'	b'	c'	d'
e'	f'	g'	h'
i'	j'	k'	l'
m'	n'	o'	p'

Random Image, R

$r_{1,1}$	$r_{2,1}$	$r_{1,2}$	$r_{2,2}$
$r_{3,1}$	$r_{4,1}$	$r_{3,2}$	$r_{4,2}$
$r_{1,3}$	$r_{2,3}$	$r_{1,4}$	$r_{2,4}$
$r_{3,3}$	$r_{4,3}$	$r_{3,4}$	$r_{4,4}$

Fig. 3. LFSR keystream distribution for random image generation considering a toy image example of size 4 × 4.

	1	2	3	4	⋯	$b-1$	b
1	$r_{1,1}$	$r_{2,1}$	$r_{1,2}$	$r_{2,2}$	⋯	$r_{1,\frac{b}{2}}$	$r_{2,\frac{b}{2}}$
2	$r_{3,1}$	$r_{4,1}$	$r_{3,2}$	$r_{4,2}$	⋯	$r_{3,\frac{b}{2}}$	$r_{4,\frac{b}{2}}$
3	$r_{1,b/2+1}$	$r_{2,b/2+1}$	$r_{1,b/2+2}$	$r_{2,b/2+2}$	⋯	$r_{1,b}$	$r_{2,b}$
4	$r_{3,b/2+1}$	$r_{4,b/2+1}$	$r_{3,b/2+2}$	$r_{4,b/2+2}$	⋯	$r_{3,b}$	$r_{4,b}$
⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮
$b-1$	$r_{1,\left(\frac{b^2}{4}-\frac{b}{2}\right)+1}$	$r_{2,\left(\frac{b^2}{4}-\frac{b}{2}\right)+1}$	$r_{1,\left(\frac{b^2}{4}-\frac{b}{2}\right)+2}$	$r_{2,\left(\frac{b^2}{4}-\frac{b}{2}\right)+2}$	⋯	$r_{1,\frac{b^2}{4}}$	$r_{2,\frac{b^2}{4}}$
b	$r_{3,\left(\frac{b^2}{4}-\frac{b}{2}\right)+1}$	$r_{4,\left(\frac{b^2}{4}-\frac{b}{2}\right)+1}$	$r_{3,\left(\frac{b^2}{4}-\frac{b}{2}\right)+2}$	$r_{4,\left(\frac{b^2}{4}-\frac{b}{2}\right)+2}$	⋯	$r_{3,\frac{b^2}{4}}$	$r_{4,\frac{b^2}{4}}$

Fig. 4. Distribution of LFSR keystream for generating random image for an image of size $b \times b$.

6.4. Random image generation

The generation of random image used to blind the original image depends on the processing. While computing single level Haar DWT on image shares, linear combination of adjacent 4 pixels takes place as shown in Fig. 2. Therefore in order to retain the randomness provided by the LFSR keystream in the share after DWT computation, it is required to use random symbols from four linearly independent keystreams to blind the four adjacent pixels, which are being linearly combined. The steps for generating linearly independent LFSR keystreams is already given in Algorithm 1. Now, the distribution of the four keystreams for generating the random images used to blind the original image is illustrated in Fig. 3 with a toy example consisting of image with size 4 × 4. In the figure, $r_{1,1}, r_{1,2}, r_{1,3}, r_{1,4}$ is the keystream generated using LFSR initial state, k_1. Similarly, k_2, k_3 and k_4 generates other keystreams $r_{2,i}$, $r_{3,i}$ and $r_{4,i}$ for $1 \le i \le 4$.

In general, the distribution of different keystreams for generating the random image used to blind an image of size $b \times b$ is shown in Fig. 4. Here $r_{i,1}, r_{i,2}, \ldots, r_{i,b^2/4}$ is the keystream generated using LFSR key, k_i, where $i = 1, 2, 3, 4$.

The proposed scheme can be extended to image decomposition using l-level Haar DWT. During single level Haar DWT computation, four adjacent pixels are linearly combined. Therefore, 4^l pixels are linearly combined during l-level Haar DWT computation. Hence random images should be generated using 4^l linearly independent LFSR initial states.

From the single level Haar DWT decomposition demonstrated in Fig. 2, it is clear that DWT computation involves pixel level operations.

Hence n shares are generated corresponding to each image pixel in (t, n) MSSS scheme, as the secret data is assigned as the constant term of the polynomial used for share generation. Whereas in (t, n) MRSS scheme, n shares are generated corresponding to a group of t image pixels. Hence in order to facilitate the required processing operation, care must be taken while arranging the t pixels used to generate the shares. In other words, the pixels to be linearly combined during SD processing should not be grouped together to form t pixels which generates a share. For instance, in the 4 ×4 image I shown in Fig. 2, the pixels b', e', f' cannot be taken as the coefficient term in MRSS along with the pixel, a' as these pixels are linearly combined during single level Haar DWT computation.

7. Security analysis

In this section, security of proposed schemes is first analyzed under collusion of t-1 participants. Then the mathematical cryptanalysis is also done to evaluate the security of the proposed schemes against known plaintext attack through collusion of threshold number of participants. Finally, the security of the proposed schemes against different statistical attacks are analyzed through simulation studies. The security and performance is analyzed by carrying out image decomposition based on single-level Haar DWT in SD.

7.1. Security against collusion of t-1 participants

In general, (t, n) secret sharing schemes provide information theoretic security, which implies that no information about the secret can be retrieved from knowledge of (t-1) shares even if the adversary has infinite computational capabilities. Secret sharing schemes can be classified as perfect [20] and/or ideal [21] based on the information theoretic conditions. Let (t, n) be a secret sharing scheme where the n shares, $f(x_1), f(x_2), \ldots, f(x_n)$ of secret s is shared among n participants. Then the (t, n) secret sharing scheme is said to be perfect if the following conditions are satisfied.

1. The secret s can be reconstructed from any t shares

$$H\left(s \mid f(x_1), f(x_2), \ldots, f(x_t)\right) = 0 \tag{29}$$

2. No information about s can be obtained from the knowledge of any $t-1$ of fewer shares

$$H\left(s \mid f(x_1), f(x_2), \ldots, f(x_{t-1})\right) = H(s) \tag{30}$$

For the above 2 conditions to hold, it is necessary that $H\left(f(x_i)\right) \geq H(s)$, $i = 1, 2, \ldots, n$. This implies that for a secret sharing scheme to be perfect, the size of share, $\left|f(x_i)\right|$ should be at least equal to the size of secret, $|s|$, i.e., $\left|f(x_i)\right| \geq |s|$, $i = 1, 2, \ldots, n$.

Now, a secret sharing scheme is said to be ideal if the information rate, $\rho = 1$. Here, information rate, ρ is defined as,

$$\rho = \frac{log_2 |S|}{log_2 |F|} \tag{31}$$

where $|S|$ is the set of possible secrets and $|F|$ is the set of possible shares. Hence, for a secret sharing scheme to be ideal, the size of the secret should be equal to the size of share.

The proposed MSSS is a modified version of Shamir secret sharing scheme where the size of the secrets are same as that of shares since their domains are equal. Therefore, the (t, n) proposed MSSS scheme satisfies perfectness and ideality conditions defined above. But the proposed (t, n) MRSS scheme is non-perfect and non-ideal since the size of shares are lesser than that of secrets.

In general, ramp schemes try to reduce the share size by relaxing the perfect secrecy conditions. The notion of secrecy for ramp schemes is computational secrecy which implies that no computational information about the secret can be gained from the knowledge of (t-1) shares by computationally bounded adversaries [26]. In (t, n) ramp scheme, the secret is a vector of size t. The secret could be any one out of p^t different vectors as the shares are computed over finite field, F_p. If the adversary knows (t-1) shares and their indices, he can retrieve the secret by guessing the tth share. If it is a wrong guess, the resulting secret differs from the original secret in all t positions [42]. Hence, the probability of retrieving the secret by an adversary who knows (t-1) shares and their indices is $\frac{1}{p}$ since there are p possibilities for the unknown share. However, the proposed MRSS scheme resists the above mentioned attack as the indices are hidden from the cloud servers which stores the share. Even though, the adversary manages to obtain (t-1) shares, he will not be able to retrieve the secret with a probability $\frac{1}{p}$ since the indices are unknown. Moreover, the secret is also blinded with keystreams of LFSR. So, the computational complexity of the proposed MRSS scheme against adversary possessing (t-1) shares will be definitely higher than that for the known-plaintext attack through collusion of t participants, detailed in the next section.

7.2. Known-plaintext Attack (KPA) during collusion of t participants

An attacker can mount a KPA in order to retrieve the secret keys if he possess ciphertexts and some plaintexts. In KPA against the proposed MSSS and MRSS scheme through collusion of t CSPs, the attacker tries to retrieve the Lagrange bases with the knowledge of t secrets and their corresponding shares. Since the random numbers are designed in such a way that their effect cannot be canceled during linear combinations, it is impossible to find the Lagrange bases by removing the contribution of random number from the set of t equations. Then the attacker can retrieve the key only through guessing the random number and then trying to find γ_j or $\gamma_{j,i}$, $1 \leq j \leq t$ by solving the set of t equations. As a result, a successful attack requires following steps.

(1) Pick t secrets, s_i, $1 \leq i \leq t$ in MSSS and t sets of secrets, $s_{i1}, s_{i1}, \ldots, s_{it}$, $1 \leq i \leq t$ in MRSS and their corresponding shares, y_{ij}, $1 \leq i, j \leq t$.
(2) Guess one set of initial keys k_1, β_1 and feedback polynomial, $q(z)$, which are the secret keys corresponding to LFSR random number generator.
(3) In MSSS, from t equations between pairs (s_i, y_{ij}), solve for γ_j using Eq. (10). While in MRSS, from t sets of equations between pairs (s_{ij}, y_{ij}), solve for γ_{ji}.
(4) Repeat steps (2) and (3) for all possible values of keys k_1, β_1 and $q(z)$ and create the table corresponding to possible solutions of γ_j or γ_{ji}.
(5) Pick a new secret and its share.
(6) Solve for γ_j or γ_{ji} using all the possible solutions of γ_j or γ_{ji} from the table.

The key space of random image generation system includes all possible combinations of LFSR initial state, k_1, feedback polynomial, $q(z)$ and multiplier key, β_1. The feedback polynomial, $q(z)$ used in LFSR should be primitive and the number of such polynomials of degree L over F_p is $\phi(p^L - 1)/L$, where ϕ denotes the Euler totient function. The number of initial states, k_1 (see Theorem 2) is given by $\prod_{j=1}^{r}(p^{d_j} - 1)$, where d_j is the degree of $f_j(z)$ which corresponds to the irreducible factors of $z^L - 1$ [43]. The number of possibilities for β_1 is given by $(p-1)^{L-1}$. Hence, the number of possible combinations of LFSR based random number generator is given by $\phi(p^L - 1)/L \cdot \prod_{j=1}^{r}(p^{d_j} - 1) \cdot (p-1)^{L-1}$. So in step (4), the attacker has $\phi(p^L - 1)/L \cdot \prod_{j=1}^{r}(p^{d_j} - 1) \cdot (p-1)^{L-1}$ possible combinations for γ_j or γ_{ji} and he has to try all these possibilities for γ_j or γ_{ji} in step (6). Therefore, the keyspace for the KPA is $2 \cdot \phi(p^L - 1)/L \cdot \prod_{j=1}^{r}(p^{d_j} - 1) \cdot (p-1)^{L-1}$. The field size, p should be at least 257, since the pixel values of the image varies from 0 to 255. The length of LFSR, $L \geq 4$ for single level Haar DWT computation. Therefore by considering a field size $p = 257$ and LFSR length $L = 4$, the keyspace is approximately 2^{83} and average computational complexity in mounting a successful KPA is at least 2^{82}. It can be seen that the attack complexity

increases to 2^{134} when L increases to 8, with the same field size $p = 257$. It should be noted that this level of security is independent of the threshold number of shares, t. Hence it is possible to have a secure MSSS and MRSS scheme even with two shares. Therefore $(2, 2)$ MSSS and MRSS schemes are considered in simulation studies.

7.3. Statistical attacks

Magnetic resonance imaging (MRI) systems are widely used for pathological brain detection. DWT based feature extraction is widely employed for disease diagnosis and for the classification of pathological brains and healthy brains using MRI. Hence, for simulations we have used healthy brain MRI and Alzheimer's disease brain MRI datasets from the website of Harvard university [44]. Simulation studies to evaluate the performance of the proposed schemes are done on PC with Intel(R) Xeon(R) CPU E3-1226 v3 3.3 GHz 16GB RAM running on Windows 10 Professional equipped with MATLAB R2015b environment. The evaluations of both MSSS and MRSS schemes are carried out by creating 2 shares corresponding to MR images of 256 × 256 size and using single level Haar DWT.

Even though field size, $p = 257$ is sufficient, in order to preserve the computation accuracy in SD, the field size must be large enough to hold the result of processing. While performing single level Haar DWT computation, the blinded MR images are first preprocessed by multiplying with 2; and 4 image pixels are linearly combined during single level Haar DWT computation. Since the final value of the decomposed pixels on reconstruction could be positive or negative, it is necessary to select p at least twice larger than the range of values. Thus to ensure the correctness of image decomposition using single level Haar DWT in SD, all the modulo operations need to be carried out using a prime number at least $\geq 255 \times 4 \times 4$. In this work, all simulations for image decomposition are carried out using $p = 4091$.

The resistance to statistical attacks is validated through histogram analysis, key sensitivity analysis and correlation analysis.

7.3.1. Histogram analysis

A good secret sharing scheme should generate a uniformly distributed histogram corresponding to the image shares. This will prevent the attacker from obtaining any information about the original image from the histogram of the image shares. To demonstrate this, the normal brain MRI and its shares along with their histograms are shown in Fig. 5. Since the shares generated using both MSSS and MRSS schemes are similar, only one set is shown in figure. The figure shows that the histograms of image shares generated through proposed schemes are completely different from those of original image and is uniformly distributed. Hence the proposed schemes are secure against histogram attack.

7.3.2. Key sensitivity analysis

Key sensitivity of proposed scheme is analyzed by comparing shares generated with keys which are differing only in a few bits. A good secret sharing scheme should produce totally different shares. Key sensitivity is usually expressed in terms of unified average changing intensity (UACI) and number of pixels change rate (NPCR) [45]. The UACI and NPCR values for various MR images are shown in Table 4. A cryptographic scheme provides high key sensitivity if UACI value lies in the range 33.3% to 33.8% and NPCR value is above 99.5% [45]. The values in Table 4 proves that the proposed schemes exhibit high key sensitivity.

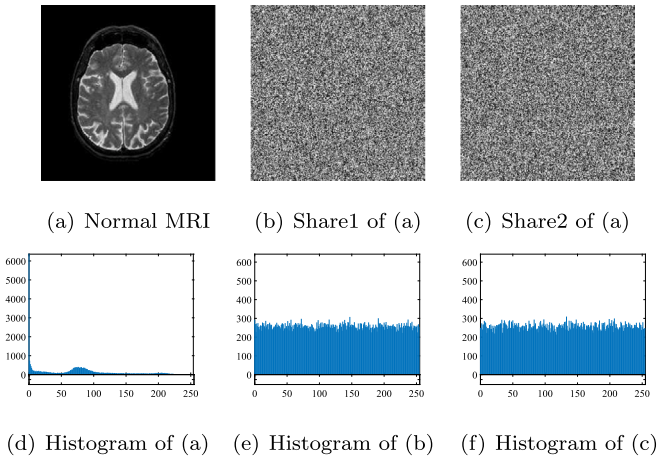

Fig. 5. MR image, its shares (S1 & S2) and their corresponding histograms.

Table 4
Key sensitivity analysis.

MR Images	UACI	NPCR
Image 1	33.32%	99.99%
Image 2	33.34%	99.99%
Image 3	33.40%	99.99%
Image 4	33.32%	99.99%

Table 5
Correlation coefficients of MR image in plaintext domain (PD) and SD; and decomposed MR image in SD.

Direction	MR Image			Decomposed MR	
	PD	SD-S1	SD-S2	SD-S1	SD-S2
Horizontal	0.9662	0.0013	−0.0028	−0.0023	0.0046
Vertical	0.9705	−0.0025	0.0073	0.0034	−0.0081
Diagonal	0.9454	−0.0043	0.0057	0.0012	−0.0053

7.3.3. Correlation analysis

In general, each pixel of an image will be highly correlated with its adjacent pixels either in horizontal, vertical or diagonal directions. This correlation among image pixels can be quantified in terms of correlation coefficient and these values should be low in order to withstand statistical attacks which exploits the correlation among adjacent pixels. Table 5 shows the correlation analysis of plaintext MR image and its shares before and after decomposition. It is evident from Table 5 that the proposed scheme fully randomizes the image pixels and no information is leaked from the image shares.

8. Performance analysis

In this section, the accuracy of SD decomposition is analyzed through qualitative and quantitative evaluation. The performance of the proposed schemes in terms of storage overhead and computational complexity is also evaluated by comparing with related schemes.

8.1. Qualitative analysis

The MR image and its two shares before and after decomposition are shown as (a) - (e) in Fig. 6. The reconstructed DWT image from its decomposed shares before and after removing random image are shown as (f) and (g); and the DWT image obtained in PD is shown as (h) in Fig. 6. From (g) and (h) in Fig. 6, it is clear that the SD DWT computation gives same accuracy as that in PD.

The qualitative analysis of SD image decomposition for four different MR images (normal and Alzheimer's brain MRI) shown in Fig. 7 indicates that the SD results matches with the PD results.

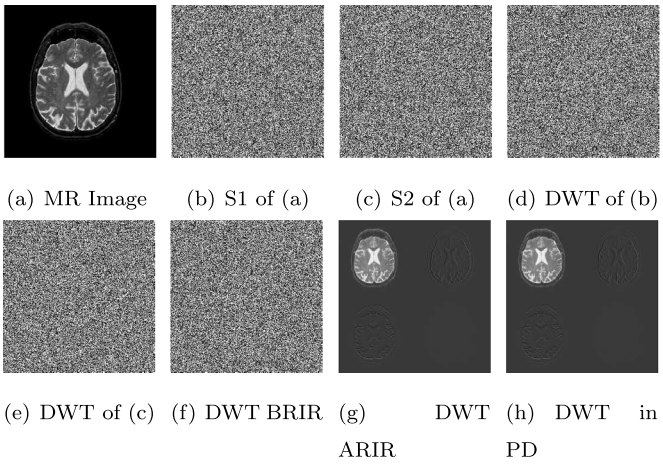

Fig. 6. Original MR image, their shares (S1 & S2) before and after decomposition; decomposed shares; reconstructed image from decomposed shares before and after removal of random image and decomposed image in PD. Here before and after random image removal are abbreviated as BRIR and ARIR respectively.

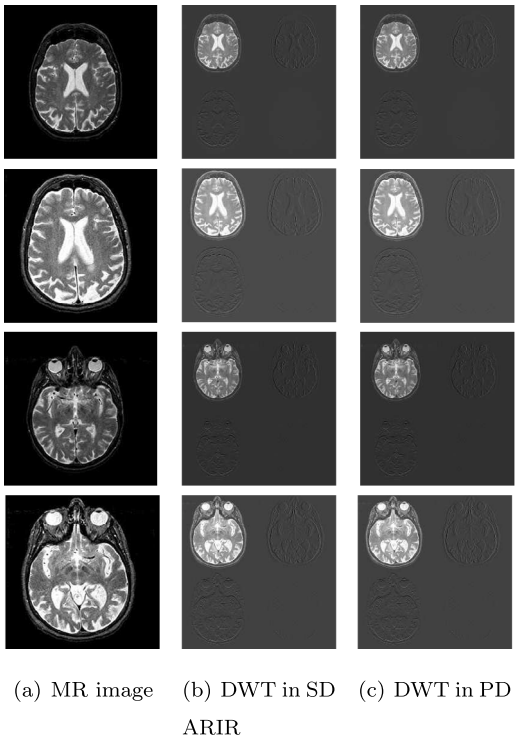

Fig. 7. MR images and their corresponding decomposed images in SD after reconstruction; and in PD.

8.2. Quantitative analysis

The accuracy of the SD image decomposition is first analyzed in terms of various performance metrics such as structural similarity index (SSIM), normalized correlation coefficient (NCC), mean-square error (MSE), normalized absolute error (NAE), maximum difference (MD), average difference (AD), structural content (SC) [46] and image quality index (IQI) [47]. Table 6 shows the values of these metrics corresponding to different images. SSIM, SC and NCC values corresponding to the images are equal to 1; IQI values are also close to 1 and MSE, NAE, MD and AD values for all images are equal to zero, which indicates that reconstructed version of SD decomposed images are very close to the PD decomposed images.

Table 6
Quantitative evaluation results of the decomposed image in SD.

Image	SSIM	SC	NCC	IQI	MSE	NAE	MD	AD
Image 1	1.0000	1.0000	1.0000	0.9936	0.0000	0.0000	0.0000	0.0000
Image 2	1.0000	1.0000	1.0000	0.9812	0.0000	0.0000	0.0000	0.0000
Image 3	1.0000	1.0000	1.0000	0.9857	0.0000	0.0000	0.0000	0.0000
Image 4	1.0000	1.0000	1.0000	0.9873	0.0000	0.0000	0.0000	0.0000

Table 7
Quantitative Evaluation Results based on decomposed images in PD and SD.

Image	H		Std. D	
	PD	SD	PD	SD
Image 1	3.0026	3.0026	57.21	57.21
Image 2	4.0338	4.0338	99.55	99.55
Image 3	3.2888	3.2982	57.93	57.93
Image 4	4.3889	4.3905	99.76	99.76

The closeness of SD image decomposition is also compared with that of PD decomposition in terms of metrics such as entropy (H) and Standard Deviation (Std. D) [46]. Table 7 shows the H and Std. D values of PD and SD image decomposition for different images. It should be noted from Table 7 that the shared domain results are very close to the plaintext domain results.

8.3. Storage overhead

The storage overhead for the proposed MSSS scheme can be expressed in terms of the number of shares being outsourced to the cloud. In the proposed (2, 2) MSSS and MRSS scheme, since only two shares can provide a security of more than 80 bits, the storage overhead will be double in MSSS, while there is no storage overhead in MRSS scheme. Whereas in the case of original (t, n) SSS and RSS scheme, the storage overhead is n times and n/t times respectively.

8.4. Computational complexity

The computational complexity of the proposed schemes can be expressed in terms of number of additions and multiplications. The client side computational complexity involves the share generation process before outsourcing and reconstruction of original processed data after retrieving processed shares. The share generation step involves one modular addition during blinding, one modular multiplication during pre-processing; and one modular multiplication and one modular addition in the generation of each share. Thus the overall computational complexity of the share generation step per image pixel includes 3 modular multiplications and 3 modular additions in F_p for proposed (2, 2) MSSS and MRSS scheme, which involves generation of two shares.

Similarly the reconstruction of the processed secret involves 2 modular multiplications for generation of Lagrange bases; and 2 modular multiplications and one modular addition in F_p for Lagrange interpolation. This is followed by post processing and removal of random number which includes one modular multiplication and one modular subtraction in F_p respectively. So the overall complexity of the secret reconstruction step is 5 modular multiplications and 2 modular additions in F_p, considering the complexity of addition is same as subtraction. The bit complexity of modular multiplication and modular addition operation are $O((logp)^2)$ and $O(logp)$ respectively. Thus, the computational complexity of share generation and secret reconstruction process for the proposed (2, 2) MSSS and MRSS scheme is $O(3\times(logp)^2)$ bit multiplications (BM) and $O(3 \times logp)$ bit additions (BA); and $O(5 \times (logp)^2)$ BM and $O(2 \times logp)$ BA respectively.

8.5. Comparison and discussions

In this section, the proposed scheme is compared with secret sharing scheme based on Chinese remainder theorem (CRT) which also possess additive homomorphism [23,37]; Elliptic Curve–ElGamal (EC–EG) based additive HES [48] and Paillier additive HES [15], which are widely used for ED computation.

8.5.1. Secret sharing scheme based on CRT

The most widely used secret sharing scheme based on CRT is Asmuth–Bloom scheme [23], whose additive homomorphic property is utilized for secure domain processing [37]. For (t, n) threshold secret sharing scheme based on CRT, $n + 1$ prime numbers are required and shares are generated using congruence equations in CRT. Let the sequence of pairwise coprime integers be $m_0 < m_1 < m_2 < \cdots < m_n$ such that $\prod_{j=1}^{t} > m_0 \prod_{j=1}^{t-1} m_{n-j+1}$; and the secret be s_1 such that $s_1 < m_0$. For share generation, first pick a random integer, A such that $s_1 + A.m_0 < \prod_{j=1}^{t} m_j$. Then the n shares, $y_j, 1 \le j \le n$ are generated as,

$$y_j = (s_1 + A.m_0) \ mod \ m_j \tag{32}$$

The coprimes, $m_j, 1 \le j \le n$ will act as the key for retrieving secret from any of the t shares. Thus the secret, s_1 can be recovered using shares, $y_1, y_2, \ldots, y_t$ by solving the set of congruence equations. By CRT, since $m_1, m_2, \ldots, m_t$ are pairwise coprime, the system has a unique solution over modulo M, where $M = \prod_{j=1}^{t} m_j$. The secret s_1 can be recovered by first solving the system of congruence equations, followed by reduction modulo m_0 as shown in Eq. (33) and (34).

$$(s_1 + A.m_0) = \sum_{j=1}^{t} \left[y_j . \left(\frac{M}{m_j} \right) . \left(\left(\frac{M}{m_j} \right)^{-1} mod \ m_j \right) \right] \ mod \ M \tag{33}$$

$$s_1 = (s_1 + A.m_0) \ mod \ m_0 \tag{34}$$

For fair comparison, the proposed modifications such as blinding the secret and hiding the coprime integers (keys) are applied to (2, 2) secret sharing scheme based on CRT. Therefore, the share generation step involves one modular addition during blinding, one modular multiplication during preprocessing; and one modular multiplication and one modular addition in share generation. Thus the overall computational complexity of the share generation step per image pixel involves 3 modular multiplications and 3 modular additions for generation of two shares. Similarly, the reconstruction of secret from two shares involves two modular division for $\left(\frac{M}{m_j} \right), j = 1, 2$, two modular inversion for $\left(\frac{M}{m_j} \right)^{-1}, j = 1, 2$, two modular multiplications and one modular addition for finding $(s_1 + A.m_0)$ from Eq. (33). This is followed by post processing and removal of random number which includes one modular multiplication and one modular subtraction respectively. So the overall complexity of secret reconstruction involves 7 modular multiplications and 2 modular additions, since the bit complexity of modular inversion and modular division is same as that of modular multiplication.

The bit complexity of share generation and secret reconstruction is computed based on modulo m_0 since the secret size depends only on m_0 (i.e., $s_1 < m_0$). But it is to be noted that the original complexity will be greater as share generation and secret reconstruction are done based on higher modulo values. Thus, the computational complexity of share generation and secret reconstruction process for the (2, 2) secret sharing scheme based on CRT is $O(3 \times (log\ m_0)^2)$ bit multiplications (BM) and $O(3 \times log\ m_0)$ bit additions (BA); and $O(7 \times (log\ m_0)^2)$ BM and $O(2 \times log\ m_0)$ BA respectively. Also, as the modulus involved in each share generation is different and is greater than m_0, the size of share will also be higher than the size of secret, which makes it a non-ideal secret sharing scheme.

8.5.2. Elliptic curve ElGamal (EC–EG) scheme

The security of EC–EG scheme relies on elliptic curve discrete logarithm problem (EC DLP). In EC–EG scheme [49], an elliptic curve over finite field, $F_{p'}$ need to be selected and p' should be a prime number of length at least 160-bit in order to provide 80-bit security. In EC–EG scheme, the plaintext is first mapped to a point on the elliptic curve and the ciphertext is represented as two curve points. A point on an elliptic curve consists of an x co-ordinate and a y co-ordinate. But only x co-ordinates of each point and sign of the y co-ordinates will be taken to form the ciphertext as it is possible to derive the y co-ordinates from the elliptic curve equation. As a result, the ciphertext size is approximately double that of the plaintext size, which will result in storage overhead.

In EC–EG scheme [49], p' of 163-bits is used. Encryption requires one point addition and two scalar multiplications on 163-bit elliptic curve. Double and add algorithm is usually used for scalar multiplication and it requires p' doublings and $p'/2$ additions. Thus the encryption process demands on an average 495 (= 2.(163 + 82)) point doublings and additions as p' is a 163-bit prime number. Each point addition and doubling operation involves approximately 5 modular multiplications [49]. So it is equivalent to 2450 (= 495 × 5) 163-bit modular multiplications. Since the complexity of a n-bit modular multiplication is $O(n^2)$, the complexity of encryption of a plaintext is $O(2450.(163^2))$. Similarly, the decryption needs only one scalar multiplication and thus the decryption complexity involves 1225 163-bit modular multiplications. Thus, the number of bit multiplications (BM) and number of bit additions (BA) involved in the encryption and decryption of a plaintext in this scheme can be mathematically expressed as,

$$BM_{Enc} = 2450 \cdot (log_2 p')^2 \tag{35}$$

$$BA_{Enc} = (log_2 p') \tag{36}$$

$$BM_{Dec} = 1225 \cdot (log_2 p')^2 \tag{37}$$

The main problem with EC–EG scheme is that in order to retrieve the plaintext s, mapping function need to be reversed. This reverse mapping function is done using pollard-rho method [50]and equivalent to solving the discrete logarithm problem (DLP) on elliptic curve. This is a brute-force approach of finding a solution to the DLP and if $0 \le s \le T$, then the expected time to solve DLP is $O(T^{1/2})$ [49]. Hence the decryption complexity depends on the plaintext size to be retrieved and if the plaintext size is large, it cannot be recovered from elliptic curve point due to the hardness of elliptic curve DLP.

8.5.3. Paillier scheme

The security of the Paillier scheme [14] relies on the decisional composite residuosity assumption which in turn depends on the computational difficulty in integer factorization. The plaintexts are represented as elements of $Z_{q'}$, where $Z_{q'}$ denotes the set of integers modulo q'. Ciphertexts are represented as an integer modulo q'^2, where q' is a product of two large primes. This means number of bits required in representing ciphertext is twice that of plaintext. In order to provide 80-bit security, q' should be of 1024 bits, which also results in huge data expansion.

The encryption process and decryption process in Paillier scheme requires 1 exponentiation and 1 modular multiplication operation. Each exponentiation needs $2 \cdot (log_2 y)$ modular multiplications, where y is the exponent. In Paillier scheme, $y \in Z_{q'}$ which implies $(log_2 y) = 1024$. Therefore, each exponentiation operation involves 2048 modular multiplications since $(log_2 y) = 1024$ as $y \in Z_{q'}$. So overall 2049 modular multiplications are involved in encryption as well as decryption process. The number of bit multiplications (BM) involved in a modular multiplication depends upon the modulus and is $(2048)^2$ as modulus is taken with respect to q'^2. Thus, the number of BM involved in the encryption and decryption of a plaintext in Paillier scheme can be obtained as,

$$BM_{Enc} = 2049 \times (2048)^2 \tag{38}$$

$$BM_{Dec} = 2049 \times (2048)^2 \tag{39}$$

8.5.4. Comparison

The comparison of the proposed secret sharing schemes with EC–EG, Paillier and CRT based secret sharing schemes are discussed in this section. Even though $p = 257$ is sufficient for providing 80-bit security, the computational complexity of proposed schemes is compared using $p = 4091$ since this value is used for SD image decomposition. In order to find the computational complexity of CRT based secret sharing scheme, $m_0 = 4091$ is used since the secret size depends on m_0. The computational complexity per image pixel and storage overhead of proposed MSSS and MRSS scheme for providing 80-bit security is compared with that of EC–EG, Paillier and CRT based secret sharing scheme in Table 8.

In the table, encryption, decryption, share generation and share reconstruction are abbreviated as Enc, Dec, Share Gen and Share Rec respectively. Since $p = 4091$ in proposed scheme and CRT based secret sharing scheme, each image pixel can be represented using 12 bits. Whereas in Paillier scheme, each image pixel is represented as an integer modulo q', where q' is a prime number with 1024 bits for providing 80 bit security. On the other hand, each image pixel is represented as a point on the elliptic curve over $F_{p'}$, where p' is a prime number with 163 bits for providing 80 bit security. Thus from Table 8, it is clear that the computational complexity of the proposed schemes and CRT based secret sharing is very less compared to EC–EG as well as Paillier scheme for comparable security levels. But, it should be noted that the computational complexity of CRT based scheme will be higher than proposed schemes since CRT scheme utilizes higher modulus values ($n + 1$ prime numbers, $m_0 < m_1 < m_2 < \cdots < m_n$) than p for share generation and secret reconstruction, whereas proposed schemes use same prime number, p for all tasks. Also, the storage overhead is double for EC–EG, Paillier and (2,2) MSSS scheme, whereas it is 1 for (2,2) MRSS scheme and more than double for CRT based scheme as the share size may be greater than the secret size. However, in EC–EG and Paillier scheme, each image pixel consisting of 8 bits is expanded to 326 and 2048 bits as the ciphertext is double the plaintext size, which will result in huge storage overhead; whereas it is expanded to only 24 bits and slightly higher in proposed MSSS scheme and CRT based scheme respectively. Even though the computational complexity and storage overhead of CRT based secret sharing scheme is almost same as that of proposed schemes, it is difficult to construct CRT based schemes since it requires a series of pairwise coprime integers satisfying some stringent conditions. The major advantage of secret sharing schemes compared to other additive homomorphic encryption schemes is that it offers data integrity and data availability. The proposed schemes can easily be extended to take advantage of original SSS by storing more number of shares than the threshold at the expense of increased storage overhead.

9. Conclusion

Shamir secret sharing (SSS) scheme is an efficient candidate for privacy preserving data processing due to its inherent additive homomorphism. In this paper, first a modified Shamir secret sharing (MSSS) scheme is proposed that resists collusion attack, which is a major drawback of original SSS scheme. This is achieved through blinding the secret before share generation. The random numbers used to blind the secrets are generated through a design based on LFSRs so that it neither spoils the homomorphic property of SSS nor destroys the randomness and security properties offered by LFSR keystream after share processing. The proposed MSSS scheme significantly reduces the storage overhead as it offers adequate security even with two shares. The data availability can be ensured by (t, n) MSSS scheme as long as any t out of n servers are available. But as the size of each share is same as that of secret, the storage overhead of MSSS scheme is high while extending it to provide data availability. Therefore a modified ramp secret sharing (MRSS) scheme is also proposed to reduce the storage overhead. The proposed (t, n) schemes can recover secret and identify the corrupted servers even when the shares in atmost $(n-t-1)$ servers are corrupted. An algorithm for image decomposition with Haar wavelet using proposed schemes is also presented. Through mathematical cryptanaysis and statistical analysis, it is proved that the proposed schemes are secure. The quantitative and qualitative analysis of simulation results showed that the shared domain processing gives same accuracy as that of plaintext domain. These attractive features make these schemes an ideal candidate for secure domain cloud applications.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

Theorem 1. The LFSR keystreams satisfy additivity and homogeneity properties.

Proof. (1) Additivity property –The addition of the LFSR keystreams will produce a new keystream, generated from an initial state which corresponds to the sum of initial states of individual keystreams.

Let $k(z) = k_0 + k_1 z + k_2 z^2 + \cdots + k_{L-1} z^{L-1}$ represent the polynomial version of LFSR initial state and $q(z) = q_0 + q_1 z + q_2 z^2 + \cdots + q_L z^L$ indicate the LFSR feedback polynomial, where $k_i, q_i \in F_p$. Then the LFSR output keystream sequence, $r(z)$ with period $p^L - 1$ can be written as,

$$r(z) = h(z)/q(z), where\ h(z) = \sum_{i=0}^{L-1} \left(\sum_{j=0}^{i} k_j q_{i-j} \right) z^i \tag{A.1}$$

Suppose $k_1(z) = k_{10} + k_{11} z + k_{12} z^2 + \cdots + k_{1L-1} z^{L-1}$ and $k_2(z) = k_{20} + k_{21} z + k_{22} z^2 + \cdots + k_{2L-1} z^{L-1}$ are two distinct LFSR initial states with the same feedback polynomial $q(z)$, then the output sequences, $r_1(z)$ and $r_2(z)$ of LFSR corresponding to $k_1(z)$ and $k_2(z)$ can be represented as,

$$\begin{aligned} r_1(z) &= \left[\sum_{i=0}^{L-1} \left(\sum_{j=0}^{i} k_{1j} q_{i-j} \right) z^i \right] / q(z) \\ r_2(z) &= \left[\sum_{i=0}^{L-1} \left(\sum_{j=0}^{i} k_{2j} q_{i-j} \right) z^i \right] / q(z) \end{aligned} \tag{A.2}$$

Then,

$$\begin{aligned} r_1(z) + r_2(z) &= \left[\sum_{i=0}^{L-1} \left(\sum_{j=0}^{i} (k_{1j} + k_{2j}) q_{i-j} \right) z^i \right] / q(z) \\ &= \left[\sum_{i=0}^{L-1} \left(\sum_{j=0}^{i} k_{3j} q_{i-j} \right) z^i \right] / q(z) \\ &= r_3(z) \end{aligned} \tag{A.3}$$

where $r_3(z)$ is the keystream obtained using the LFSR initial state, $k_3(z) = k_1(z) + k_2(z)$. □

Proof. (2) Homogeneity property –The scalar multiplication of an LFSR keystream will produce a new keystream, generated from an initial state which corresponds to the scalar multiple of the initial state of original keystream.

Let $r(z)$ denote the keystream generated using initial key, $k(z)$. Then $r'(z) = \alpha \cdot r(z)$ is the keystream obtained using $k_\alpha(z) = \alpha \cdot k(z)$, where $\alpha \in F_p$.

$$\begin{aligned} \alpha \cdot r(z) &= \alpha \cdot \left[\sum_{i=0}^{L-1} \left(\sum_{j=0}^{i} k_j q_{i-j} \right) z^i \right] / q(z) \\ &= \left[\sum_{i=0}^{L-1} \left(\sum_{j=0}^{i} \alpha \cdot k_j q_{i-j} \right) z^i \right] / q(z) \end{aligned}$$

Table 8
Comparison of the proposed (2, 2) MSSS scheme with EC–EG and Paillier scheme for 80-bit security level.

Parameters	EC–EG [49]	Paillier [15]	(2, 2) MSSS & MRSS Scheme, CRT based Secret Sharing [37]
Computational Complexity	Enc - 2^{26} BM + 2^{8} BA Dec - 2^{25} BM	Enc - 2^{33} BM Dec - 2^{33} BM	Share Gen - 2^{9} BM + 2^{6} BA Secret Rec - 2^{10} BM + 2^{5} BA
Storage Overhead	2	2	2 (MSSS) 1 (MRSS) > 2 (CRT based Secret sharing)

$$= \left[\sum_{i=0}^{L-1}\left(\sum_{j=0}^{i} k_{\alpha j} q_{i-j}\right) z^i\right] / q(z)$$
$$= r'(z) \quad \square \tag{A.4}$$

Appendix B

Theorem 2. The set of L initial states of an LFSR of length L derived through cyclic shift of the initial secret key $k(z)$ will be linearly independent, if gcd $(k(z), z^L - 1)$ forms a polynomial of zero degree.

Proof. Let the LFSR initial state (secret key) be denoted as $k = (k_0, k_1, k_2, \ldots, k_{L-1})$, where $k_i \in F_p$. Also define a shifting operator, $D : W \mapsto W$ as,

$$D(k_0, k_1, k_2, \ldots, k_{L-1}) = (k_{L-1}, k_0, k_1, \ldots, k_{L-2}) \tag{B.1}$$

Now, arranging the secret key, k and its corresponding $L-1$ shifted versions as the rows of the matrix, K', an $L \times L$ circulant matrix can be formed as shown in Eq. (B.2).

$$K' = \begin{bmatrix} k \\ Dk \\ \vdots \\ D^{L-2}k \\ D^{L-1}k \end{bmatrix} = \begin{bmatrix} k_0 & k_1 & \cdots & k_{L-2} & k_{L-1} \\ k_{L-1} & k_0 & \cdots & k_{L-3} & k_{L-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ k_2 & k_3 & \cdots & k_0 & k_1 \\ k_1 & k_2 & \cdots & k_{L-1} & k_0 \end{bmatrix} \tag{B.2}$$

If the circulant matrix, K' is of full rank, then every initial state of LFSR generated by cyclically shifting the coefficients of $k(z)$ will be linearly independent.

Let the entries of a circulant matrix T are,

$$T_{ij} = \begin{cases} 1, j - i \equiv 1 (mod\ L) \\ 0, j - i \not\equiv 1 (mod\ L) \end{cases} \tag{B.3}$$

Now, the $L \times L$ circulant matrix K' which correspond to the secret key, k can be represented as $K' = \sum_{i=0}^{L-1} k_i T^i$ where $T^0 = I$ is an $L \times L$ identity matrix and T^1 is obtained through the cyclic shift of each row of T^0 using the operator D. Then, there exists a circulant matrix $C = \sum_{i=0}^{L-1} c_i T^i$, where $c_i \in F_p$ such that $K'.C = I$, if K' is invertible over F_p.

If $k(z) = \sum_{i=0}^{L-1} k_i z^i$ is the polynomial representation of circulant matrix, K', then computing the inverse of K' is equivalent to finding a polynomial $c(z) = \sum_{i=0}^{L-1} c_i z^i$ in $F_p[z]$ such that,

$$k(z).c(z) \equiv 1 (mod(z^L - 1)) \tag{B.4}$$

The congruence relationship in Eq. (B.4) is due to the equality $T^L = I$. With the help of Extended Euclidean algorithm, Eq. (B.4) can also be written as,

$$k(z).c(z) + n(z).(z^L - 1) = 1 \tag{B.5}$$

Thus from Eq. (B.5), it is clear that the circulant matrix, K' of size $L \times L$ will have full rank if gcd $(k(z), (z^L - 1)) = u$, where u is a non-zero integer in F_p. In general, the rank of circulant matrix is $L - g$, if gcd $(k(z), (z^L - 1))$ is having a degree g. Hence, if $g = 0$, then all rows of K' are linearly independent and it will be a full rank matrix. $\square$

References

[1] Ali M, Khan SU, Vasilakos AV. Security in cloud computing: Opportunities and challenges. Inf Sci 2015;305:357–83.
[2] Singh S, Jeong Y-S, Park JH. A survey on cloud computing security: Issues, threats, and solutions. J Netw Comput Appl 2016;75:200–22.
[3] Mthunzi SN, Benkhelifa E, Bosakowski T, Guegan CG, Barhamgi M. Cloud computing security taxonomy: From an atomistic to a holistic view. Future Gener Comput Syst 2020;107:620–44.
[4] Dimakis AG, Godfrey PB, Wu Y, Wainwright MJ, Ramchandran K. Network coding for distributed storage systems. IEEE Trans Inform Theory 2010;56(9):4539–51.
[5] Rodrigues R, Liskov B. High availability in dhts: Erasure coding vs. replication. In: International workshop on peer-to-peer systems. Springer; 2005, p. 226–39.
[6] Lakshmi VS, Deepthi PP. A secure regenerating code-based cloud storage with efficient integrity verification. Int J Commun Syst 2019;32(9):e3948.
[7] Liu X, Deng RH, Yang Y, Tran HN, Zhong S. Hybrid privacy-preserving clinical decision support system in fog–cloud computing. Future Gener Comput Syst 2018;78:825–37.
[8] Wang X, Bai L, Yang Q, Wang L, Jiang F. A dual privacy-preservation scheme for cloud-based ehealth systems. J Inf Secur Appl 2019;47:132–8.
[9] Zhang C, Zhu L, Xu C, Lu R. Ppdp: An efficient and privacy-preserving disease prediction scheme in cloud-based e-healthcare system. Future Gener Comput Syst 2018;79:16–25.
[10] Liang P, Zhang L, Kang L, Ren J. Privacy-preserving decentralized abe for secure sharing of personal health records in cloud storage. J Inf Secur Appl 2019;47:258–66.
[11] Bouslimi D, Coatrieux G, Cozic M, Roux C. A joint encryption/watermarking system for verifying the reliability of medical images. IEEE Trans Inf Technol Biomed 2012;16(5):891–9.
[12] Alloghani M, Alani MM, Al-Jumeily D, Baker T, Mustafina J, Hussain A, et al. A systematic review on the status and progress of homomorphic encryption technologies. J Inf Secur Appl 2019;48:102362.
[13] Dossogne J, Lafitte F. Blinded additively homomorphic encryption schemes for self-tallying voting. J Inf Secur Appl 2015;22:40–53.
[14] Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of the EUROCRYPT, vol. 99. Springer; 1999, p. 223–38.
[15] Zheng P, Huang J. Discrete wavelet transform and data expansion reduction in homomorphic encrypted domain. IEEE Trans Image Process 2013;22(6):2455–68.
[16] Jiang L, Xu C, Wang X, Luo B, Wang H. Secure outsourcing sift: Efficient and privacy-preserving image feature extraction in the encrypted domain. IEEE Trans Dependable Secure Comput.
[17] Shamir A. How to share a secret. Commun ACM 1979;22(11):612–3.
[18] Benaloh JC. Secret sharing homomorphisms: Keeping shares of a secret secret. In: Conference on the theory and application of cryptographic techniques. Springer; 1986, p. 251–60.
[19] Blakley GR. Safeguarding cryptographic keys. In: International workshop on managing requirements knowledge. IEEE; 1979, p. 313–8.
[20] Van Dijk M. On the information rate of perfect secret sharing schemes. Des Codes Cryptogr 1995;6(2):143–69.
[21] Brickell EF, Davenport DM. On the classification of ideal secret sharing schemes. J Cryptol 1991;4(2):123–34.
[22] Mignotte M. How to share a secret. In: Workshop on cryptography. Springer; 1982, p. 371–5.
[23] Asmuth C, Bloom J. A modular approach to key safeguarding. IEEE Trans Inf Theory 1983;29(2):208–10.
[24] Goldreich O, Ron D, Sudan M. Chinese remaindering with errors. In: Proceedings of the thirty-first annual ACM symposium on theory of computing. 1999. p. 225–34.
[25] Quisquater M, Preneel B, Vandewalle J. On the security of the threshold scheme based on the chinese remainder theorem. In: International workshop on public key cryptography. Springer; 2002, p. 199–210.
[26] Krawczyk H. Secret sharing made short. In: Annual international cryptology conference. Springer; 1993, p. 136–46.
[27] Goryczka S, Xiong L. A comprehensive comparison of multiparty secure additions with differential privacy. IEEE Trans Dependable Secure Comput 2015;14(5):463–77.
[28] Bogdanov D, Laur S, Willemson J. Sharemind: A framework for fast privacy-preserving computations. In: European symposium on research in computer security. Springer; 2008, p. 192–206.

[29] Bogdanov D, Niitsoo M, Toft T, Willemson J. High-performance secure multi-party computation for data mining applications. Int J Inf Secur 2012;11(6):403–18.
[30] Bogdanov D, Talviste R, Willemson J. Deploying secure multi-party computation for financial data analysis. In: International conference on financial cryptography and data security. Springer; 2012, p. 57–64.
[31] Burkhart M, Strasser M, Many D, Dimitropoulos X. Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. Network 1(101101).
[32] Dolev S, Gupta P, Li Y, Mehrotra S, Sharma S. Privacy-preserving secret shared computations using mapreduce. IEEE Trans Dependable Secure Comput.
[33] SaghaianNejadEsfahani SM, Luo Y, Cheung S-cS. Privacy protected image de-noising with secret shares. In: 19th IEEE international conference onimage processing. IEEE; 2012, p. 253–6.
[34] Mohanty M, Ooi WT, Atrey PK. Scale me, crop me, knowme not: Supporting scaling and cropping in secret image sharing. In: IEEE international conference on multimedia and expo. IEEE; 2013, p. 1–6.
[35] Lathey A, Atrey PK. Image enhancement in encrypted domain over cloud. ACM Trans Multimedia Comput Commun Appl 2015;11(3):38.
[36] Singh P, Agarwal N, Raman B. Secure data deduplication using secret sharing schemes over cloud. Future Gener Comput Syst 2018;88:156–67.
[37] Singh P, Raman B. Reversible data hiding for rightful ownership assertion of images in encrypted domain over cloud. AEU-Int J Electron Commun 2017;76:18–35.
[38] Xiang Y, Xiao D, Wang H, Li X. A secure image tampering detection and self-recovery scheme using pob number system over cloud. Signal Process 2019;162:282–95.
[39] Dautrich JL, Ravishankar CV. Security limitations of using secret sharing for data outsourcing. In: IFIP annual conference on data and applications security and privacy. Springer; 2012, p. 145–60.
[40] Menezes A, Van Oorschot P, Vanstone S. Handbook of applied cryptography. CRC Press; 1996.
[41] Gong X, Hu P, Shum KW, Sung CW. A zigzag-decodable ramp secret sharing scheme. IEEE Trans Inf Forensics Secur 2018;13(8):1906–16.
[42] McEliece RJ, Sarwate DV. On sharing secrets and reed-solomon codes. Commun ACM 1981;24(9):583–4.
[43] MacWilliams FJ. Orthogonal circulant matrices over finite fields, and how to find them. J Combin Theory Ser A 1971;10(1):1–17.
[44] Summers D. Harvard whole brain atlas: www. med. harvard. edu/aanlib/home. html. J Neurol Neurosurg Psychiatry 2003;74(3):288.
[45] Wu Y, Noonan JP, Agaian S. Npcr and uaci randomness tests for image encryption, Cyber journals: multidisciplinary journals in science and technology. J Sel Areas Telecommun 2011;1(2):31–8.
[46] Blum RS, Liu Z. Multi-sensor image fusion and its applications. CRC Press 2005.
[47] Wang Z, Bovik AC. A universal image quality index. IEEE Signal Process Lett 2002;9(3):81–4.
[48] Li L, Abd El-Latif AA, Niu X. Elliptic curve elgamal based homomorphic image encryption scheme for sharing secret images. Signal Process 2012;92(4):1069–78.
[49] Mykletun E, Girao J, Westhoff D. Public key based cryptoschemes for data concealment in wireless sensor networks. In: IEEE international conference on communications, vol. 5. IEEE; 2006, p. 2288–95.
[50] Pollard JM. Monte carlo methods for index computation (mod p). Math Comput 1978;32(143):918–24.