Contents lists available at ScienceDirect

Journal of Information Security and Applications

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

Design of compressed sensing fault-tolerant encryption scheme for key sharing in IoT Multi-cloudy environment(s)

Peng Zhang[a], Juntao Gao[a,*], Wenjuan Jia[a], Xuelian Li[b]

[a] State Key Laboratory of Integrated Services Networks, China
[b] School of Mathematics and Statistics, Xidian University, Xi'an, Shaanxi, China

ARTICLE INFO

Keywords:
Compressive sensing
Internet of Things
Cloud storage security
Fault tolerance
Authentication and key exchange
Energy protection

ABSTRACT

The booming growth of the Internet of things(IoT) has improved the productivity of companies and improved people's quality of life. However, as a growing number and variety of connected devices are introduced into the IoT networks, not only are communication bandwidth and storage severely challenged, but potential security threats are escalating. Therefore, we consider using compressive sensing(CS) in the IoT system to solve this problem. The solution we designed solves three main security threats in the IoT, the perception layer, the transport layer and the application layer. In our work, the gateway node of the perception layer provides the function of anonymous identity authentication and information authentication, which can not only prevent external attackers from obtaining private information from the gateway node, but also solve security problems such as identity authentication and man-in-the-middle attack in the transmission layer. Meanwhile, our scheme handles the ciphertext integrity protection and energy leakage in existing CS encryption(CSE) schemes, protecting the security of application layer privacy data. And our scheme provides a secure key sharing method, which improves the security of the system. In addition, we designed two fault-tolerant models to ensure that the ciphertext can still be decrypted correctly when the cloud server fails in the IoT environment. The use of the replaced sparse vector instead of the random matrix method in the traditional CSE scheme improves the quality of recovery while saving a large amount of storage space. Simulation results show that our scheme improves the overall compression and recovery performance compared to other CSE schemes.

1. Introduction

A recent study by Hewlett Packard found that 70% of the most commonly used Internet of Things(IoT) devices [15] have serious vulnerabilities due to lack of transport encryption and insufficient authorization. Armis, an IoT security research firm, found eight zero-day vulnerabilities in the bluetooth protocol that will affect more than 5.3 billion devices in 2017 - from Android, iOS, Windows and Linux systems devices to IoT devices using short-range wireless communications technology. In July 2018, Microsoft introduced its highest-level Windows bug reward program to date, up to $250,000, to address the various vulnerabilities that have arisen in the Internet of Things. And the McKinsey Global Institute says IoT has the potential to create economic impact between $2.7 trillion to $6.2 trillion annually by 2025.

As shown in Fig. 1, the IoT system has been applied in many fields such as intelligent transportation, environmental protection, smart medical care, smart city, etc., and has greatly promoted people's daily life. However, as more and more connected devices are introduced into the IoT network [16], these hidden authentication, access management, privacy leakage and storage security threats are increasingly prominent. These security threats mainly come from three aspects: perception layer, transmission layer and application layer. The perception layer security threat mainly comes from the leakage of the key of the gateway node in the wireless sensor network. The transport layer security threat refers to the security issue of identity authentication in different network structures. The application layer security threat comes from the leakage of large amounts of user privacy data. Therefore, many identity-based authentication protocols [17–19] have been proposed to address the security threats in the IoT. These protocols can partially address security issues such as identity authentication and privacy protection. However, as the data stream of users' personal information sharing becomes larger and larger, many IoT applications need massive data storage, huge processing speed to realize real-time decision-making, and high-speed broadband network to transmit data. Moreover, users have put forward higher requirements for IoT security and privacy protection of personal information.

* Corresponding author.
E-mail address: jtgao@mail.xidian.edu.cn (J. Gao).

https://doi.org/10.1016/j.jisa.2019.04.004
2214-2126/© 2019 Published by Elsevier Ltd.

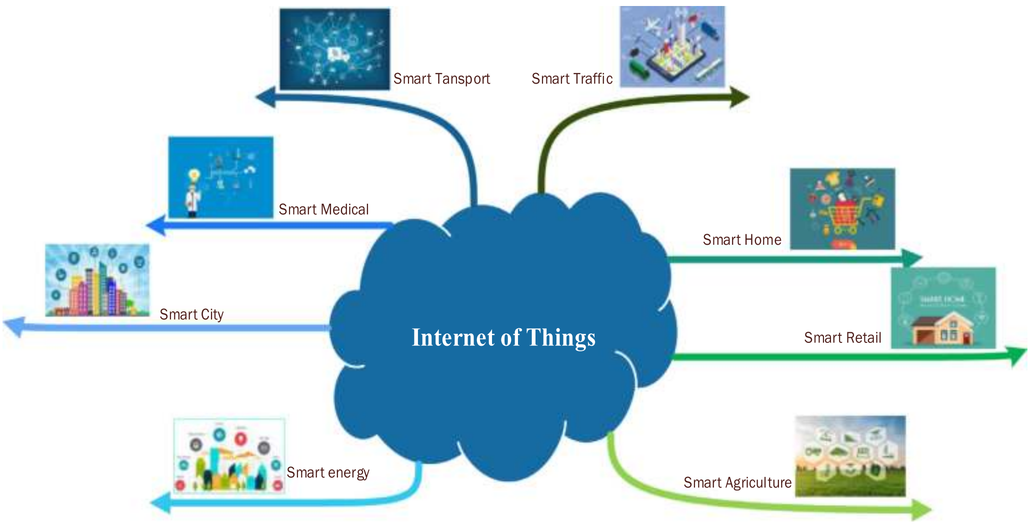

Fig. 1. The application of IoT system.

Cloud computing provides an ideal back-end solution for processing large volumes of data. Many electronic transactions [23] based on public cloud systems also have more and more applications. So people consider combining cloud computing with the IoT to improve the efficiency of storage and computing. However, Cisco reports that, by the end of 2018, data center traffic will increase to 10.8 ZB per year (1ZB= GB), and cloud system traffic will account for more than 76% of the total data center traffic, and this figure was 54% last year. By 2020, 92% of the global workload will be handled by cloud data centers. In order to solve the pressure of high-speed bandwidth, we consider using CS to solve the bandwidth problem of massive data. Compressive sensing (CS) [1–3] is a novel theory framework for signal description and processing. CS allows accurately or approximately to reconstruct the original signal with much lower sampling rate than the Nyquist sampling theorem, thus addressing the problem of insufficient bandwidth when transferring huge amount of data.

1.1. Related works

CS itself can provide certain security protection for the transmission and storage of the data information [9]. In work [4], the theory of using measuring matrix as secret key is studied. In [5], Rachlin and Baron first formally study measurement matrix in 2008. They indicated that the perfect secrecy in CS-based encryption modes is unreachable but the computational secrecy can be achieved, since the signal after compression retains the energy information of the original signal. However, using the same measurement matrix to encrypt multiple different signals might result in serious security issues. And Gao in [7] generates the measurement matrix by the pseudo random generator to ensure CS security. Therefore, it is indicated in [6,8] that key seed is used to generate different measurement matrix through LFSR. In [10,11] CS-processed data are uploaded to the cloud in order to save resources and ensure security. Xue proposes a cloud encryption scheme based on CS, which supports the function of random compression, statistical decryption and accurate decryption [12]. And Zhang et al. proposed a method to replace random matrix by permutation in [13,14]. And it is suitable for the transmission of data in lossy channels in the muti-cloudy environment. However, once the lost ciphertext packet exceeds half of the total data, the recovery effect cannot be guaranteed.

The works mentioned above to some extent use CS to achieve the security of encrypted data. However, these solutions cannot achieve user authentication and key sharing. Moreover, none of these papers takes the issue of ciphertext integrity protection into account. Therefore, it will cause serious security problems in practical. They are not able to be used in IoT systems where there are many security challenges.

1.2. Our contributions

To address the problems above, we propose design of compressed sensing fault-tolerant encryption scheme for key sharing in IoT multi-cloudy environment(s). Fig. 2 shows the overall system framework. In our paper, CS provides encryption and compression of transmitted data. This enables the data stored in the application layer of the IOT system cryptographic state. Therefore, it can guarantee the security of user privacy data in the IoT system. Meanwhile, CS's good compression feature solves the problem of massive data storage. At the same time, we design the anonymous identity authentication and information authentication scheme for the IOT system, which solves the security problems such as identity authentication attacks and man-in-the-middle attacks faced by the transport layer. Finally, the gateway node of the sensing layer only provides the authentication function. Even if an external attacker attacks the gateway, it cannot obtain any key information, and the security of the user key is guaranteed.

Therefore, the work in this paper not only solves the three major threats in the Internet of Things system, but also solves the problem of excessive bandwidth transmission of massive data, improves the security performance of the IOT system and saves storage space. To summarize, our main contributions are as follows:

- For the CS-based IoT cloud system, we designed two fault-tolerant models, which can set the appropriate fault-tolerant scheme according to the probability of the failure of cloud server in the IoT system, making the data transmission more reliable. Thereby improving the fault tolerance performance of the IoT system, and is suitable for the actual application model.
- Aiming at the security risks of the perception layer in the IoT system, anonymous identity authentication and information authentication are designed in the gateway node, and the key exchange between users is realized. It proves that the designed user authentication and key exchange protocol can resist various attacks and improve the security of the gateway node.
- CS-based ciphertext authentication and energy encryption are designed. Energy standardization makes the energy of the encrypted data indistinguishable from the energy of the original

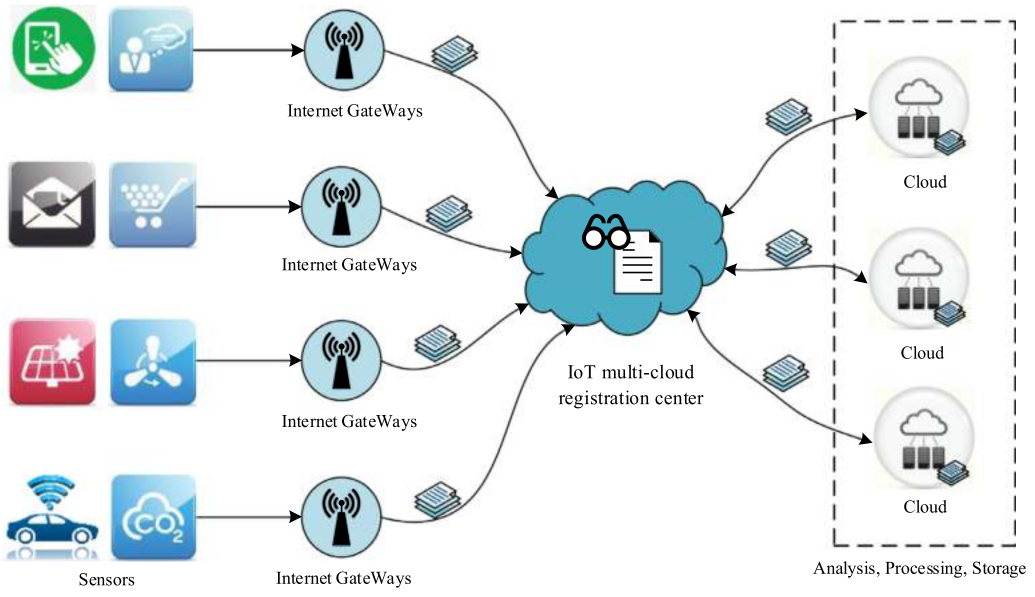

Fig. 2. The overall system framework.

data. Making the ciphertext stored in the application layer more secure. The simulation results show that compared with other CS encryption schemes, our scheme has better recovery effect while providing stronger security.

- Saving a lot of storage space. By setting the appropriate threshold, only high-dimensional vector is saved after sparseness, and then the replacement vector is used to replace the random matrix in the traditional CS encryption. Not only improves the recovery performance of the solution, but also greatly saves the storage space (the sparse vector space saves at least two-thirds).

The rest of the paper is organized as follows. Section II introduces compressive‘ sensing briefly. We provide the detailed scheme in Section III, and system analysis in Section IV. Section V gives our experimental simulation. Finally, Section VI draws a conclusion remarks.

2. Compressive sensing

CS can realize the transformation from the original high-dimensional signal to the low-dimensional signal while maintaining the necessary information to reconstruct the original signal. By solving the sparse optimization problem, the conversion from the low-dimensional signal to the original high-dimensional signal can be achieved. Let $x \in \mathbb{R}^n$ denote an original signal. Then x is called sparse or compressible if there exists a basis $\Psi \in \mathbb{R}^{n\times n}$ such that

$$x = \Psi s \tag{1}$$

where Ψ is the sparse matrix and $s \in \mathbb{R}^n$ is a sparse vector that is the representation of x in sparse domain Ψ. Specifically, we call that the vector x is k-sparse if the sparse basis that represent x with k nonzero entries and $(n-k)$ zero or near zero entries. Let $y \in \mathbb{R}^m$ be the measurement vector, and then the random projection process is defined as:

$$y = \Phi x = \Phi\Psi s = As \tag{2}$$

where $\Phi \in \mathbb{R}^{m\times n}$ is the measurement matrix with $m < n$ and $A = \Phi\Psi$ is the sensing matrix. Through the process above, we can transform a high-dimensional signal into a low-dimensional measurement vector.

In order to obtain the original signal x from y at the decoding end, we have to solve the underdetermined Eq. (2) where the number of equations m is less than the number of variables n. So there will be more than one solution in Eq. (2). To solve the problem, Candès shows that if the sensing matrix A satisfies Restricted Isometry Property (RIP) [20], we can recover x from y by solving l_1-*norm* method:

$$\min_s \|S\|_1 \; subject\ to\ As = y \tag{3}$$

Then we can handle the problem in polynomial time. To ensure efficient compression and good recovery, we use the method in [12] to construct sparse matrices and measurement matrices. In a CS-based cloud database encryption environment, the processed ciphertext saves storage space while ensuring secure transmission.

3. Design of compressed sensing fault-tolerant encryption scheme

3.1. System architecture and process

Our system includes users, semi-honest IoT multi-cloud registration centers (all RC in the following represents semi-honest IoT multi-cloud registration center), and cloud servers. Among them, Alice and Bob represent sending data users and subscribing users respectively. The role of RC is to manage registered users and cloud servers. The cloud server is used to store and decrypt encrypted data.

Fig. 3 shows the proposed key exchange and fault tolerance processing. First, all users and cloud servers apply for registration in the RC to obtain their corresponding ID secret values. After registration, RC records the registrant's ID and the corresponding anonymous identity information. The registered user Alice uses our CS encryption method to encrypt the original plaintext. Then Alice performs fault-tolerant processing on the ciphertext and sends the processed packet data to the cloud servers. When Bob wants to subscribe the encrypted data, Bob needs perform mutual anonymous authentication with Alice through the gateway. Alice sets up the session key for the authenticated Bob and shares the decryption key with Bob using the session key. Finally, Bob sends subscription request to cloud servers. After verifying the validity of the Bob's identity, the cloud servers send the ciphertext to Bob. In our scheme, the sensing matrix A is known to the cloud servers and the complex CS decryption work is performed on the resource-rich

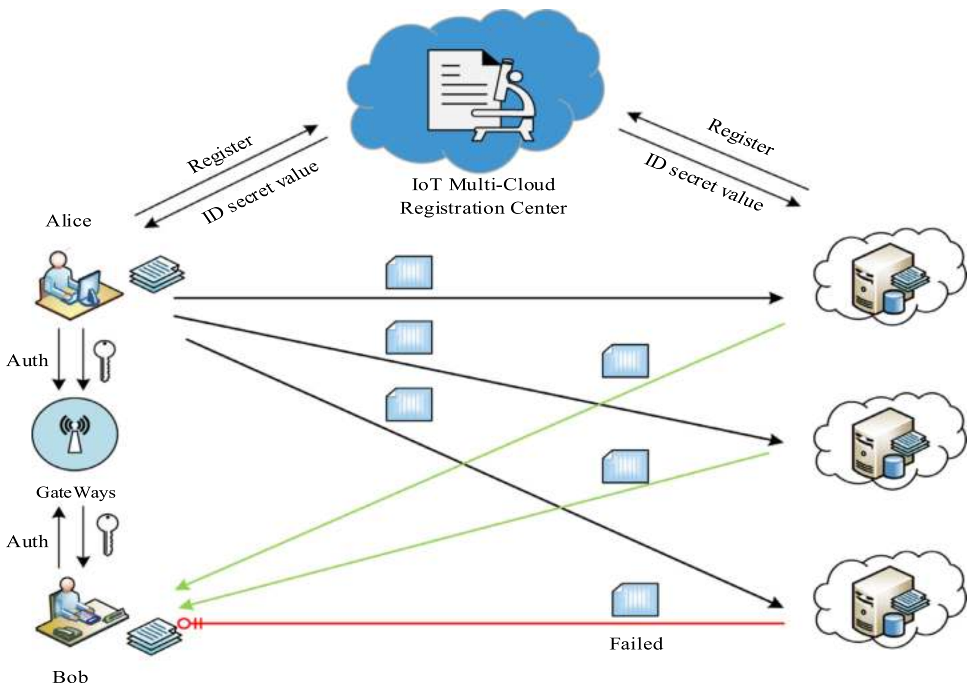

Fig. 3. The proposed key exchange and fault tolerance processing.

cloud servers side, while Bob only needs to perform simple permutation operation and energy decryption. In addition, the designed ciphertext fault-tolerant control processing method can ensure the correctness of ciphertext decryption. That is, when some of the cloud servers fail to send data, we can still recover the original plaintext using ciphertext data from the remaining cloud servers.

3.2. *Fault-tolerant control model*

We present the fault-tolerant design of muti-cloud server under two different probabilistic models, and analyze the relationship between the fault-tolerance rate α and the probability distribution under the two models. Here, we assume that the probability of each cloud server failing is independent of each other.

3.2.1. *Equal probability model*

Let the system has t independent and identically distributed cloud servers, and the probability of each cloud server failure is p.

(1) The mathematical expectation of cloud server failure is $\bar{X} = pt$, and the relationship between fault-tolerance rate and probability is $\alpha = p$.
(2) The probability of continuously i cloud servers failure is

$$P(\varepsilon = i) = C_t^i \cdot p^i \cdot (1-p)^{t-i} \tag{4}$$

and the fault-tolerance rate $\alpha = \frac{i}{t}$.

3.2.2. *Arbitrary probability model*

The number of cloud servers and the probability of failure are the same as above.

(1) The mathematical expectation of cloud server failure is $\bar{X} = p_1 + p_2 + \cdots + p_t$, and the fault-tolerance rate is $\alpha = \frac{p_1+p_2+\ldots+p_t}{t}$.
(2) The probability of continuously i cloud servers failure is (assume starting from the dth cloud server)

$$P(\varepsilon = i) = C_t^i \cdot \prod_{j=\{d,d+1,\ldots,d+i-1\}} p_j \cdot \prod_{j\neq\{d,d+1,\ldots,d+i-1\},j\in t} p_j \tag{5}$$

and the fault-tolerance rate $\alpha = \frac{i}{t}$.

3.3. *Design details*

3.3.1. *System initialization*

The semi-honest registration center RC first randomly selects two large prime numbers p, n, and generates an additive cyclic group G_1 and a multiplicative cyclic group G_2 with the same order p. Then RC randomly selects $\lambda, C \in Z_n^*$ calculates anonymous authentication function:

$$C(x) = C + \lambda \prod_{i=1}^{N} (x - s_i) \tag{6}$$

s_i represents the anonymous identity information of registered users and cloud servers. Finally, RC selects three secure hash functions H_1,H_2,H_3, pseudo-random normal distribution matrix generator Γ, and secretly keeps the secret value λ while publishes $params = \{H_1, H_2, H_3, \Gamma,\ C(x),\ C\}$ to the user group. Here, H_1 represents $G_1 \to \mathbb{Z}_n^*$, H_2 represents $G_2 \to \mathbb{Z}_n^*$, H_3represents $G_1 \times G_1 \times \{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_n^*$.

3.3.2. *Cloud server registration*

When a cloud server U_{ci} applies for registration, U_{ci} first sends identity ID_{ci} to RC. Then RC randomly selects a secret value x and calculates anonymous identity information $s_{ci} = H_2(x \cdot g^{ID_{ci}})$. g is the generator of finite field $\mathbb{Z}_n^*$. When the cloud server registration is completed, RC records $\{ID_{ci}, s_{ci}\}$ to the registered cloud server collection.

3.3.3. *User registration and self-certification*

When a user U_i applies for registration, U_i first randomly selects $k_i, pw_i \in \mathbb{Z}_n^*$ and calculates $K_i = k_i \cdot P, PW_i = g^{pw_i}$, then sends ID_i,K_2,PW_2 to RC. When receiving the user's registration request, RC randomly selects $x, r_i \in \mathbb{Z}_n^*$ and calculates $R_i = K_i + r_i \cdot P, h_i =$

$H_1(R_i), \bar{x}_i = r_i + h_i$, where x is RC's private key. And then RC calculates:

$$p_i = x \cdot g^{ID_i} \tag{7}$$

$$s_i = H_2(p_i \cdot PW_i) \tag{8}$$

Finally, RC records $\{ID_i, s_i\}$ to the registered user collection and sends $\bar{x}_i$ to registered users. After receiving the information sent by RC, the user U_i checks if the following equation can hold:

$$(\bar{x}_i + k_i) \cdot P = h_i \cdot P_{pub} + R_i \tag{9}$$

If the equation does not hold, U_i stops the session. Otherwise, U_i calculates the private key $x_i = \bar{x}_i + k_i$. Note that, through the self-certification process described above, the user can verify the legitimacy of the private key information sent by the RC. And the user's private key is generated by the RC's private key and the user's private key, preventing RC from leaking the user's private key.

3.3.4. Compressed and encrypted data

First of all, the user processes the original data with sparse matrix, and then converts the sparse data into one-dimensional(1D) data. According to the size of the 1D sparse data, the user uses the sparse matrix to generate the measurement matrix as follows:

$$\Psi = U \Lambda V^T \tag{10}$$

$$\Phi = U_m^T \tag{11}$$

Note that the sparse matrix here is generated through an over-complete learning dictionary which can find the simplest form of a signal. The sparse matrix obtained by this method has better sparseness and can achieve better reconstruction properties. The measurement matrix Φ is constructed by taking the first m columns of the orthogonal matrix U in the singular value decomposition of the sparse matrix Ψ. For more details about the construction of projection matrices, please refer to [11]. It has been shown in [21] that the measurement matrix constructed by this method has a better performance for reducing the reconstruction error.

After that, the user records the position of non-zero elements in the sparse vector and obtains the support set s_v of the sparse vector. Here, we assume that the length of the 1D sparse data is n. Then, the user randomly selects s different elements from the vector space with size n and puts them into the vector space p to obtain the permutation vector, where s represents the number of sparse vectors in 1D sparse data. Exchanging corresponding position elements in the vector space p and support set s_v as follows:

$$s(p(i)) = s(s_v(i)) \tag{12}$$

The above permutation exchange process will randomly rearrange the non-zero entries of sparse data, which is equivalent to a random perturbation matrix that makes the ciphertexts different even though the original plaintexts are same. Therefore, using the permutation exchange method can not only reduce the computational complexity but also save storage space.

Then the user use key generation parameter K_1 to compute:

$$c = \pi_{K_1}(\| \theta_i' \|_2^2) = \| \theta_i' \|_2^2 \oplus K_1 \tag{13}$$

c is the energy encrypted data, and θ' is the exchanged sparse data. The energy encryption function prevents the signal energy leakage and guarantees the signal energy security. Finally, the user uses the above encryption parameters combined with CS to encrypt the exchanged sparse data, and then use the hash function H to calculate the verification tag of the energy encryption cipher text, therefore:

$$y_i = \Phi \Psi \hat{\theta}_i' = A \frac{\theta_i'}{\| \theta_i' \|_2^2} \tag{14}$$

$$T_1 = H(y_i), T_2 = H(c_i) \tag{15a}$$

where $A = \Phi\Psi$ is the sensing matrix, and $\hat{\theta}_i'$ is the sparse data after energy encryption. The CS-based encrypted process is shown in Algorithm 1.

Algorithm 1 Compress and Encrypt Data.

```
1. Inputs: Original data $x$, sparse matrix $\Psi$, key generation parameters $K_1$,
     secure hash function $H$.
2. Generate sparse support set $s_v$ and permutation vector $p$, and exchange the
     corresponding entries in $s_v$ and $p$.
3. Calculate energy encryption function $c_i$ with and $K_1$.
4. Calculate the cipher text and authentication tag after CS encryption
     method $y = \Phi\Psi\hat{\theta}_i'$, $T_1 = H(y_i)$,$T_2 = H(c_i)$.
```

3.3.5. Ciphertext fault-tolerant control processing

The user sets the number of data groups t and fault tolerance α according to the failure probability of each cloud server. Then the user calculates the data length m and shift length k of each group

$$m = \frac{(\alpha t - t + \alpha) \cdot y}{at} \tag{15b}$$

$$k = \frac{y}{t} \tag{16}$$

Finally, the user sends $(y_i, c, T, s_v)^T$ to the multi-cloud servers, where y_i represents the grouped data after fault-tolerance processing.

3.3.6. Anonymous authentication and session key distribution

When user U_i wants to subscribe the information stored in the cloud server, U_i randomly selects $a_i \in \mathbb{Z}_n^*$ and calculates $X_i = a_i \cdot P$. And U_i uses p_i,R_i getting from RC to calculate:

$$t_i = H_2(x \cdot g^{ID_i} \cdot PW_i) \tag{17}$$

$$b_i = H_3(R_i \parallel X_i \parallel T_i \parallel vaild\ period) \tag{18}$$

Here, T_i is the user's current timestamp, and *vaildperiod* represents the authentication vaild time period. Then U_i uses the private key x_i to calculate $d_i = (a_i + b_i)^{-1} \cdot x_i$ and send $(R_i, X_i, t_i, d_i, T_i, vaildperiod)$ to user U_j. User U_j first verifies the validity of T_i and *vaildperiod* after receiving U_i data. For authenticated user U_i, U_j brings t_i to $C(x)$ for anonymous identify authentication. If $C(t_i)$ is equal to C, then U_j calculates $h_i = H_1(R_i)$,$b_i = (R_i \| X_i \| T_i \| vaildperiod)$ and verifies if $d_i(X_i + b_i P) = h_i \cdot P_{pub}$. If not equal, authentication fails. If equals, $U_j$randomly selects $a_j \in \mathbb{Z}_n^*$ and calculates $X_j = a_j \cdot P$. And U_j uses p_j,R_j getting from RC to calculate:

$$t_j = H_2(x \cdot g^{ID_j} \cdot PW_j) \tag{19}$$

$$b_i = H_3(R_j \parallel X_j \parallel T_j \parallel vaild\ period) \tag{20}$$

Here, T_j is the user's current timestamp. U_j calculates the session key$Sk_{ij} = H_1(a_j \cdot X_i)$. And U_j uses the private key x_j to calculate $d_j = (a_j + b_j)^{-1} \cdot x_j$ and send $(R_j, X_j, t_j, d_j, T_j)$ to user U_i.

When the subscriber U_i receives the U_j data, U_i first verifies the legitimacy of T_j and *vaildperiod*. For authenticated users, U_i takes t_j into $C(x)$ to verify that if $U_j's$ anonymous identity value is equal to C. If not equal, authentication fails. If equals, U_i calculates $h_j = H_1(R_j)$,$b_j = (R_j \| X_j \| T_j \| vaildperiod)$ and verifies if$d_j(X_j + b_j P) = h_j \cdot P_{pub}$. If equals, U_i calculates the session key $Sk_{ij} = H_1(a_i \cdot X_j)$. Otherwise, authentication fails.

3.3.7. Decryption key distribution and cloud server authentication

After completing the session key distribution, the user U_j encrypts the energy encryption key$K_1' = Sk_{ij} \oplus K_1$, and sends K_1' to subscriberU_i. U_i then sends the identity information t_i to the cloud servers. Cloud servers take t_i into $C(x)$ to verify that if $U_i's$ anonymous identity value is equal to C. If not equal, authentication fails. If equals, the cloud servers use the CS recovery algorithm $\hat{\theta}_i = rec(y_i, A)$ to calculate each set of energy-encrypted sparse data θ_i'. Hereafter, the cloud servers send $(\theta_i', c, T, s_v)^T$ to U_i.

3.3.8. Fault-tolerant processing

In the process of sending ciphertext data, if the previous $d-1(d-1 \le t \cdot \alpha)$ cloud servers fail, according to the designed fault-tolerance method, the user only needs to obtain $[d,\ d+(t \cdot \alpha - 1)]$ group data. Note that when $d+(t \cdot \alpha - 1) > t$, the grouped data to be acquired are $[d,\ t]$ and $[1,\ d+(t \cdot \alpha - 1)(modt)]$. Through the method above, all the information of the original data can be obtained from the partial cloud server.

In the process of sending ciphertext data, if any consecutive $d-1(d-1 \le t \cdot \alpha)$ cloud servers fail, according to the designed fault-tolerance method, the user only needs to obtain the $[d', 2d' - t]$ group data starting from the next d' of the missing data. Note that when $2d' - t > t$, the grouped data to be acquired are $[d',\ t]$ and $[1, 2d' - t(\bmod\ t)]$. Through the above method, all the information of the original data can be obtained from the partial cloud server.

3.3.9. Ciphertext decryption

The user first locally calculate the verification tag T' using hash function H by $T_1' = H(y_i), T_2' = H(c_i)$. Then the user verifies whether $T_1' = T_1$ and $T_2' = T_2$ or not. If equal, the user calculates energy encryption key $K_1 = Sk_{ij} \oplus K_1'$ and energy value of sparse data $\| \theta_i \|_2^2 = c_i \oplus K_1$ using the obtained session key Sk_{ij} and energy value c. And then the user restores the original support set $s(s_v(i)) = s(p(i))$ using the obtained per vector p. Finally, the user calculates the original date by:

$$x = \Psi \hat{\theta}_i \cdot \| \theta_i \|_2^2 \tag{21}$$

Note that permutation of the support set only changes the position of the sparse data, and the size of the sparse data is not changed. Therefore, the energy value of the sparse signal is always the same. The decryption process is shown in Algorithm 2.

Algorithm 2 Ciphertext Decryption.

```
1. Inputs: permutation vector $p$, energy encryption function $c$, session key
   $Sk_{ij}$, hash function $H$, energy encryption key $K_1'$.
2. Calculate the verification tag $T_1' = H(y_i), T_2' = H(c_i)$ and verifies whether
   $T_1' = T_1$ and $T_2' = T_2$ or not. If equal, perform the next step, otherwise
   terminate.
3. Calculate energy encryption key $K_1$ and energy value of sparse data $\| \theta_i' \|_2^2$
   with $Sk_{ij}$ and $c$ respectively.
4. Obtain the original support set $s_v$ permutation vector $p$.
5. The user calculates the raw data $x$ locally with $\Psi$ and $\| \theta_i \|_2^2$.
```

3.3.10. Revocation, recovery and add algorithms

Assume that the revoked users or cloud servers set is $\Delta = \{U_{i1}, U_{i2}, \ldots, U_{im}\}, 1 \le m \le n$, RC first removes the revocation set Δ from the anonymous authentication function $C(x)$, then selects $C', \lambda' \in \mathbb{Z}_q^*$, changes and broadcasts the anonymous authentication function $C'(x) = C' + \lambda' \prod_{i=1}^{N} (x - s_i)$ and C'. Keys and identities of other users or cloud servers remain unchanged and the plaintext can still be obtained according to the above decryption algorithm.

To recover some users or cloud servers who have been revoked, The RC only needs to add the identity information corresponding to the recovery users or the cloud servers to the anonymous authentication function $C(x)$. Unrevoked users or cloud servers do not need to update identity information.

When a new user or cloud server sends a registration request, according to the above registration process, RC adds the registered new user or cloud server's identity information to anonymous authentication function $C(x)$, while other users' keys and identity information remain unchanged.

4. System analysis

In this section, we provide detailed proof of the security of the proposed protocol in the random oracle model. The random oracle model assumes that the hash function is actually a true random function and it produces a random value for each new query.

4.1. Adversarial model

In this section, we describe the capabilities of an adversary and list security requirements of a mutual authentication and key exchange protocol. An adversary A is a probabilistic polynomial time machine. We allow A to potentially control all communications in the proposed protocol via accessing to a set of oracles as described below. Let $\alpha \in \{U, S\}$ and Π_a^m be the mth instance of α.

- Extract(ID_α): In this query model, the corresponding private key can be obtained by the identity ID_α.
- Send(Π_a^m, M): This query models that an adversary A can send a message M to the oracle Π_a^m. When Π_a^m receives M, according to the proposed protocol, it makes the computations and gives the answers.
- $H_i(m_i)$: When an adversary A makes this hash query with the message m_i, the oracle Π_a^m returns a random number r_i and records (m_i, r_i) into a list L_{H_i},where $i = 1, 2, 3$ and list is initially empty.
- Corrupt(ID_α): An adversary A can issue this query to the oracle and get back the user/server private key from RC.
- Secret(Π_a^m): With this query, the corresponding user/server secret value can be obtained by the adversary.
- Test(Π_a^m): The adversary A is allowed to send a Test query to the oracle Π_a^m. On receiving a Test query, Π_a^m chooses a bit $b \in \{0, 1\}$ and returns the session key if $b = 1$, otherwise outputs a random string $SK \in \mathbb{Z}_q^*$ as the session key.

Definition 1 (AKA-secure)**.** The authenticated key agreement (AKA) advantage is defined as

$$Adv_p^{aka}(A) = |2Pr[Asucceeds] - 1|$$

where $Pr[Asucceeds]$ *is the probability of A wins the game. We say the protocol P is AKA-secure if* $Adv_p^{aka}(A)$ *is negligible.*

In an execution of protocol $Pr[Asucceeds]$, we say adversary A violates client-to-server authentication if A can fake the authenticator $(R_i, X_i, t_i, d_i, T_i, vaildperiod)$. We denote this event and its probability by C2S and Pr[C2S] separately. Similarly, we say adversary A violates server-to-client authentication if A can fake the authenticator $(R_j, X_j, t_j, d_j, T_j)$. We denote this event and its probability by S2C and Pr(S2C) separately.

Definition 2. We say protocol P is mutual authentication-secure (MA-secure) if both Pr[C2S] and Pr[S2C] are negligible.

Theorem 1. *In our protocol, both Pr[C2C] are negligible then our scheme is MA-secure.*

Proof. If there is an adversary A can violate the client-to-client authentication, that is, Pr[C2C] is non-negligible, then we can construct an algorithm F to solve the DLP.

Given a DLP an instance (*P*, *Q*), F picks at random *I*, *J* ∈ {1, 2, ..., q_u}, guessing that A will generate a legal $U_I's$ message $(R_{U_I}, X_{U_I}, t_{U_I}, d_{U_I}, T_{U_I}, validperiod)$ to authentication with the client U_J . And *F* lets the public P_{pub} be *Q* and generates other parameters described in Section 3. Then *F* gives the system parameter $(G_1, G_2, n, C, C_x, P, P_{pub}, \mathbb{Z}_n^*, H_1, H_2, H_3)$to A, and answers A's queries as follows.

- $Extract(ID_\alpha)-queries$: *F* maintains a list L_E of form (ID_α,S_α,C_x), which is indexed by ID_α and initialized empty, where $\alpha \in U$. On receiving the queries with ID_α,K_α,PW_α, *F* looks up L_E. If ID_α is in L_E, *F* will do nothing. Otherwise, *F* generates two random numbers $a, b \in \mathbb{Z}_n^*$ and computes the registered private key

$$R_\alpha = K_\alpha + a \cdot P_{pub} + b \cdot P, \bar{x}_\alpha = b, h_\alpha = H_1(R_\alpha) \leftarrow -a \bmod n$$

$$p_\alpha = b \cdot g^{ID_\alpha}, S_\alpha = H_2(p_\alpha \cdot PW_\alpha)$$

Then *F* adds S_α to the list L_E and responds with $(R_\alpha, \bar{x}_\alpha, p_\alpha)$. It is easy to say the equation $\bar{x}_\alpha \cdot P = h_\alpha \cdot P_{pub} + R_\alpha$ holds where $x_\alpha = \bar{x}_\alpha + k_\alpha$.

- $H_3-queries$: *F* maintains verify information L_3 of form (h_α,T_α,t_α,*C*, *C*(*x*), *validperiod*).When A queries the oracle H_3 on (T_α,t_α,*validperiod*). *F* responds as follows : if T_α,*validperiod*is legal and $C(t_\alpha)=C$, then *F* responds with h_α; otherwise, *F* randomly chooses $h_\alpha \in \mathbb{Z}_n^*$ that has not been chosen by *F*, and inserts (h_α,T_α,t_α,*C*, *C*(*x*), *validperiod*) into L_3 and responds with h_α.
- $H_i-queries$: *F* maintains verify information L_i of form (m_i,h_i), which is initialized empty, where $i=1,2$. On receiving the queries, F looks up L_i. If (m_i,h_i) is on the list L_i, F responds with h_i. Otherwise, F generates a random number h_i, adds (m_i,h_i) to L_i and responds with h_i.
- $Send-queries$:
  - When adversary *A* makes a Send $(\Pi_a^m,'start')$ query to the oracle, *F* responds as follows: If $C(t_\alpha) = Candt_\alpha \neq t_{U_I}$, *F* responds with ($R_\alpha$,$X_\alpha$,$t_\alpha$,$d_\alpha$,$T_\alpha$, *validperiod*)according to the description of protocol, since *F* knows k_α, $\bar{x}_\alpha$ and x_α of the client U_α; otherwise, *F* generates a random number $t_{U_i} \in \mathbb{Z}_n^*$, computes $X_{U_i} = t_{U_i} \cdot P$ and defines $d_{U_i} \leftarrow d_{U_i}(X_{U_i} + b_{U_i}P) = R_{U_i} + h_{U_i}P_{pub}$.And then *F* responds *A* with $(R_{U_i}, X_{U_i}, t_{U_i}, d_{U_i}, T_{U_i}, validperiod)$.
  - Else *F* answers *A*'s send-query as the description of the protocol, because *F* knows the private e key and secret value of other users.
- $Corrupt-queries$: When *A* queries (ID_α,K_α,PW_α), *F* responds with $(r_\alpha, \bar{x}_\alpha, p_\alpha)$.
- $Secret-queries$: if $t_{U_i} \neq t_{U_I}$, *F* responds with the secret value d_{U_i}; otherwise, *F* cancels the game.
- $Reveal-queries$: When the adversary *A* makes a Reveal Π_a^m query, *F* responds as follows. If Π_a^m is not accepted, *F* responds with ⊥. Otherwise *F* checks L_3 and responds with the corresponding h_3.

Then, *F* can simulate the protocol perfectly under the random oracle model. Suppose *A* does outputs a legal $U_i's$ message $(R_{U_i}, X_{U_i}, t_{U_i}, d_{U_i}, T_{U_i}, validperiod)$ to authentication with the client U_J. Because *I*, *J* ∈ {1, 2, ..., q_u} then the events $i=I$ and $j=J$ happen with the same probability of $1/q_u$. Then *A* generates a legal $U_i's$ message $(R_{U_i}, X_{U_i}, t_{U_i}, d_{U_i}, T_{U_i}, validperiod)$ to authentication with the client U_J with the probability $\eta=\varepsilon/q_u q_u$ and η is non-negligible.

The tuple $(R_{U_I}, X_{U_I}, d_{U_I})$ can be considered as the signature of the message$(R_{U_I}, X_{U_I}, T_{U_I}, validperiod)$. The Forking Lemma [22] adopts the 'oracle replay attack'. If *A* can find a valid signature $(R_{U_I}, X_{U_I}, d_{U_I})$ for message ($R_{U_I}, X_{U_I}, T_{U_I}$, *validperiod*) with a non-negligible probability η, then *A* can generate another two valid message signature $(R_{U_I}, X_{U_I}, d'_{U_I})$ and $(R_{U_I}, X_{U_I}, d''_{U_I})$ with probability at least $\eta/2$ such that

$$d_{U_I}(X_{U_I} + b_{U_I} \cdot P) = R_{U_I} + h_{U_I} \cdot P_{pub}$$

$$d'_{U_I}(X_{U_I} + b'_{U_I} \cdot P) = R_{U_I} + h_{U_I} \cdot P_{pub}$$

$$d''_{U_I}(X_{U_I} + b''_{U_I} \cdot P) = R_{U_I} + h_{U_I} \cdot P_{pub}$$

where $h_{U_I} = H_1(R_i)$,d'_{U_I} and $d''_{U_I}$are two hash values from H_3. Let $X_{U_I} = a_{U_I} \cdot P$, $R_{U_I} = K_{U_I} + r_{U_I} \cdot P$ and $P_{pub} = Q$, we have

$$d_{U_I}(a_{U_I} + b_{U_I}) \cdot P = (r_{U_I} + k_{U_I}) \cdot P + h_{U_I} \cdot P_{pub}$$

$$d'_{U_I}(a_{U_I} + b'_{U_I}) \cdot P = (r_{U_I} + k_{U_I}) \cdot P + h_{U_I} \cdot P_{pub}$$

$$d''_{U_I}(a_{U_I} + b''_{U_I}) \cdot P = (r_{U_I} + k_{U_I}) \cdot P + h_{U_I} \cdot P_{pub}$$

In these equations, only a_{U_I},r_{U_I},$k_{U_I}$are unknown to *F*. *F* solves these values from the above three linear independent equations, and outputs k_{U_I}as the solution of the discrete logarithm problem. Therefore, *F* can solve the DLP with a non-negligible probability. Therefore, Pr(C2C) is negligible and the proposed protocol can provide the client-to-client authentication. Through the same method, we can prove that Pr(C2C) is negligible and the proposed protocol can provide the client-to-client authentication if the DLP is hard.

Theorem 2. *Assume that an adversary A can guess the value b involved in the Test-query with a non-negligible advantage ε. Then we can construct an algorithm F from A to solve the computational Diffie-Hellman problem. Therefore, our scheme is AKA-secure.*

proof. Suppose that an adversary *A* can win the game described in previous Section with non-negligible probability ε. Then, we can use the ability of *A* to construct an algorithm *F* solving the CDH problem.

Let q_U,q_{se}, and q_h be the total number of user, number of session and number hash queries. Suppose *F* is challenged with a CDH problem instance$P_1 = aP$, $P_2 = bP$ and is tasked to compute *abP*, where *F* does know the value of *a*, *b*. *F* picks $x \in \mathbb{Z}_n^*$, l_1,l_2 ∈ {1, ..., q_{Se}}, *I*, *J* ∈ {1, ..., q_U}at random, and gives (G_1,G_2,*n*, *C*, C_x,*P*, $P_{pub}=xP, Z_n^*, H_1, H_2, H_3$) to *A* as the public parameters. Then *F* answers *A*'s queries as follows.

- $Extract(ID_\alpha)-queries$: Oracle responses similar to Theorem 1.
- $H_3-queries$: *F* maintains verify information L_1 of form (h_α,T_α,t_α,*C*, *C*(*x*), *valid period*). When *A* queries the oracle H_3 on (T_α,t_α,*valid period*). *F* responds as follows : if T_α,*validperiod*is legal and $C(t_\alpha)=C$, then *F* responds with h_α; otherwise, *F* randomly chooses $h_\alpha \in \mathbb{Z}_n^*$ that has not been chosen by *F*, and inserts (h_α,T_α,t_α,*C*, *C*(*x*), *validperiod*) into L_1 and responds with h_α.
- $H_i-queries$: *F* maintains verify information L_1 of form (m_i,h_i), which is initialized empty, where $i=1,2$. On receiving the queries, *F* looks up L_i. If (m_i,h_i) is on the list L_i, F responds with h_i. Otherwise, *F* generates a random number h_i, adds (m_i,h_i) to L_i and responds with h_i.
- $Send-queries$:
  - if adversary *A* makes a Send query $(\Pi_a^m = \Pi_{U_i}^{l_1})$ and $M=\lambda$, *F* generates a random number $a_{U_i} \in \mathbb{Z}_n^*$, computes $X_{U_i} = a_{U_i} \cdot P$ and defines $d_{U_i} \leftarrow d_{U_i}(X_{U_i} + b_{U_i}P) = R_{U_i} + h_{U_i}P_{pub}$. And then *F* responds *A* with $M_1 = (R_{U_i}, X_{U_i}, t_{U_i}, d_{U_i}, T_{U_i}, validperiod)$.
  - Else, if $(\Pi_a^m = \Pi_{U_j}^{l_2})$ and $M=M_1$, *F* checks if the equation$d_{U_i}(X_{U_i} + b_{U_i} \cdot P) \ = R_{U_i} + h_{U_i} \cdot P_{pub}$ and $C(t_{U_i}) = C$holds. If the equation does not hold, *F* stops the session. Otherwise, generates a random number $a_{U_j} \in \mathbb{Z}_n^*$, computes $X_{U_j} = a_{U_j} \cdot P_2$ and defines $d_{U_j} \leftarrow d_{U_j}(X_{U_j} + b_{U_j}P) =$

$R_{U_j} + h_{U_j} P_{pub}$. And then F responds A with $M_2 = (R_{U_j}, X_{U_j}, t_{U_j}, d_{U_j}, T_{U_j}, validperiod)$.

- Else, if $(\Pi_a^m = \Pi_{U_i}^{l_1})$ and $M = M_2$, F checks if the equation$d_{U_j}(X_{U_j} + b_{U_j} \cdot P) = R_{U_j} + h_{U_j} \cdot P_{pub}$ and $C(T_{U_j}) = C$holds. If the equation does not hold, F stops the session. Otherwise, accepts the session.
- Else F answers A's Send queries as the description of the protocol, because F knows the private e key and secret value of other users.

– *Corrupt – queries*: When A queries ID, F responds to the corresponding registration information.

– *Secret – queries*: if $t_{U_i} \neq t_{U_I}$ or $t_{U_j} \neq t_{U_J}$, F responds with the secret value d_{U_i} or d_{U_j}; otherwise, F cancels the game.

– *Reveal – queries*: When adversary A makes a $\Pi_{U_i}^m$ query, F responds as follows. If $\Pi_{U_i}^m$ is not accepted or the session key of $\Pi_{U_I}^m$ and $\Pi_{U_J}^m$ is queried, F responds with $\perp$. Otherwise F checks L_3 and responds with the corresponding h_3.

– Test-query: if the query is not between $\Pi_{U_I}^{l_1}$ and $\Pi_{U_J}^{l_2}$, then F cancels the game; otherwise F flips an unbiased coin b and returns the session key if $b = 1$, otherwise outputs a random string $SK \in \mathbb{Z}_n^*$ as the session key.

Simulation of F is perfectly indistinguishable from the proposed protocol excepting in the following cases: $d_{U_I} \leftarrow d_{U_I}(X_{U_I} + b_{U_I}P) = R_{U_I} + h_{U_I}P_{pub}$ and $d_{U_J} \leftarrow d_{U_J}\ (X_{U_J} + b_{U_J}P) = R_{U_J} + h_{U_J}P_{pub}$ cannot be done in the Send queries, respectively. Indeed, b_{U_I} and b_{U_J} are two values from H_3, d_{U_I} and d_{U_J} are two elements from $\mathbb{Z}_n^*$. This simulation with the failure probability is less than $\frac{2q_{se}q_h}{nq_U}$.

The probability that A chooses $\Pi_{U_I}^{l_1}$ as the Test oracle is $1/q_U q_{se}$. In this case, A would not have made Reveal $\Pi_{U_I}^{l_1}$ or Reveal $\Pi_{U_J}^{l_2}$ queries, and so A would not have aborted. If A can win in such a game, then A must have made the corresponding H_1 query of the form $H_1(K_{ij})$. if $\Pi_{U_I}^{l_1}$ is the initiator oracle with overwhelming probability because H_1 is a random oracle. Thus, A can find the corresponding item in the H_1-list with the probability $1/q_h$ and output $(a_{U_I}a_{U_J})^{-1}K_{IJ}$ as the solution to the CDH problem. Therefore, if the adversary computes the correct session key with non-negligible probability ε, then the probability that F solves the CDH problem is $(\varepsilon/q_U q_{se} q_h) - (2q_{se}q_h/nq_U)$ which is non-negligible. Therefore, $Adv_p^{aka}(A)$ is negligible and the proposed protocol can provide the AKA-security.

Through the similar method, we can prove our scheme could provide forward secrecy property. We describe it as the following theorem.

Theorem 3. *Our protocol has the perfect forward secrecy property if the CDH problem is hard.*

Proof. According to Theorem 2, the session key in our protocol depends not only on the registered private key but also on the secret value of the user. On the other hand, we prove in Theorem 2 that our protocol is still secure even if the registered private key is leaked. Therefore, our protocol can offer perfect forward security. In addition, our scheme withstands the following possible attacks for multi-server environment.

Replay attack: In our scheme, replaying a message of previous session into new session is invalid, because the user's in the mutual authentication generate different nonce values $T_i andvaildperiod$ in each new session, which make all messages dynamic and valid for the session only.

Malicious user attack: Although the client user U_i can compute the secret key x_i by $\bar{x}_i$ and k_i, he/she cannot derive x and r_i from $\bar{x}_i$ because he/she ca not solve the DLP.

Malicious server attack: In our solution, the server can't impersonate other servers to attack, because each server needs to register identity information in the RC. Once it pretends to be another server, it cannot pass the identity information verification.

Impersonation attack: In the proposed scheme, the adversary Eve may construct a valid login message from the previous login message (R_i,X_i,t_i,d_i), where the secret key k_i is embedded in R_i. However, Eve cannot derive the secret key k_i from R_i on the security of CDLP. Thus, Eve cannot construct a valid message without knowing k_i. In this case, the user forgery attack will fail.

Known session key security: We call a protocol is session key security, if an adversary knowing some previous session keys cannot get the session keys of the current round in the key agreement protocol. In our protocol, the agreed session key relies on the one-way hash function $h(.)$ and secrets values selected by mutual authentication users. The output of the hash function is distributed uniformly in $\{0, 1\}^k$. In other words, one session key is independent with the others. Besides, the session secrets are generated with the random value a_i,a_j. Thus, even one session's secrets are revealed, the other session secrets will remain safe.

User reparability: In practical applications, if the client operator cannot log in to the service server by submitting the identity and password corresponding to the mobile device, the access is denied. When this happens, the user only needs to re-select the PW_i and send it to the RC for registration, and the user's real identity ID does not need to be changed. Therefore, our solution can complete the re-registration of the user identity without changing the user's real ID in this situation.

5. Experimental simulation

We conduct a simulation test on the Intel® Core(TM) i3-6100 CPU @ 3.70 GHz in DELL. We use one-dimensional cosine signals and two-dimensional image signals to simulate the scheme. The simulation is mainly carried out in three aspects: time, space and recovery quality under different compression ratios. The results illustrate that the proposed compressed sensing fault-tolerant encryption scheme for key sharing is secure and feasible.

First of all, we use the cosine function to test the security performance of our scheme. To be more representative, we select four cosine functions with different frequencies and amplitudes for the simulation test. Note that in the experimental simulation, the tested the signal length is 120 and the compression ratio is 0.5. The four sets of simulation results in Fig. 4 correspond to four different cosine functions. The first one on the left represents the original signal figure, the second represents the encrypted signal figure, the third represents the comparison of the original signal with the decrypted signal, the fourth represents the comparison of the unsubstituted decryption signal with the decrypted signal, and the fifth represents the comparison of the decrypted signal with the original signal. Note that due to the inherent nature of compressed sensing, we can maintain good recovery by saving only large projected sparse values. In order to balance the loss of recovered signal quality and storage space, we set the appropriate projection threshold. After simulation, we can recover the cosine signal of length 120 with high quality by saving only 6 sparse values.

By comparing the original signal with the encrypted signal, we can see that the signal after the encryption has been completely distorted. So even if a malicious attacker could get the ciphertext, he cannot find any similarity or association with the plaintext. In order to further prove the importance of the energy encryption process in our scheme, we give the third column chart which shows the comparison of the recovered decrypted signal and the recovered signal without energy decryption. Obviously, we can see from the comparison that the recovery signal without decryp-

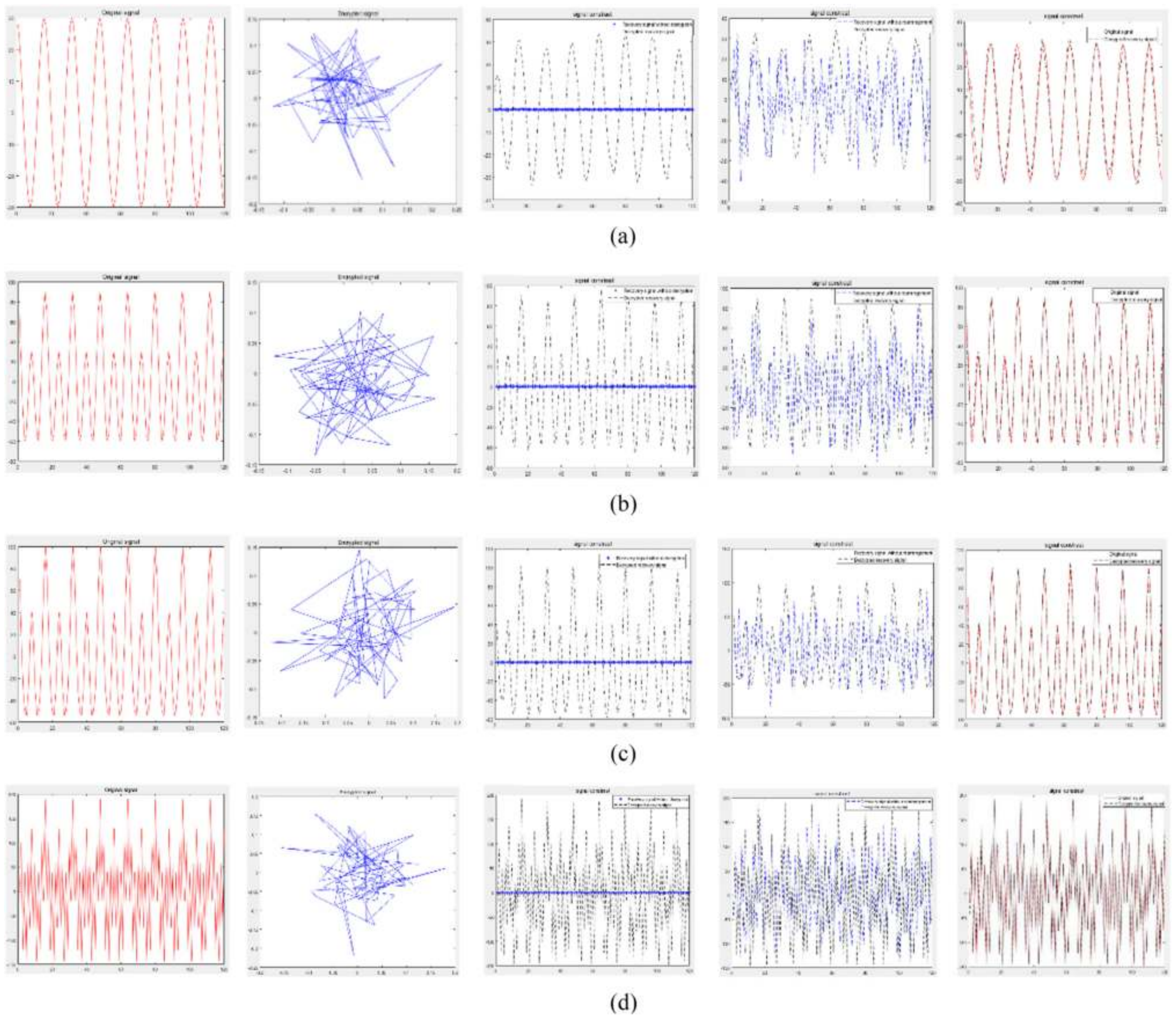

Fig. 4. Simulation comparison of different curves.

tion is completely different from the original signal. Therefore, if the ciphertext is not decrypted, no information of the original data can be obtained even if the malicious eavesdropper obtains the CS-recovered signal. This is because after energy normalization, the energy of each signal is protected. If the energy decryption operation is not performed, the recovered signal values are nearly zero. From the fourth column, we can see that the unsubstituted decrypted signal is completely distorted. From the fifth column, we can find that the signal curve recovered basically coincides with the original signal curve. This result shows that our program has very good recovery effect. Therefore, our scheme can achieve good data security performance.

Table 1 shows the relative error of the above four cosine curve at different compression ratios, and it is the average result from 100 simulation tests. From the results in Table 1, it can be seen that when the compression ratio is 0.5, the relative error of all the recovery curves is controlled within 5%. Moreover, when the compression ratio is 0.6, the simulation test results show that the best relative error is only 0.1%. This means that our solution can maintain good recovery characteristics, which is extremely important in practical applications. More encouragingly, the relative error of the recovery curve decreases as the complexity of the test curve increases. This shows that our scheme has better applicability in testing huge amounts of data, and it is very suitable in IOT and cloud encryption system.

Table 1
Relative error of different curves under different compression.

CR	0.2	0.3	0.4	0.5	0.6
(a)	0.1264	0.0998	0.0733	0.0505	0.0305
(b)	0. 0533	0.0463	0.0207	0.0112	0.0064
(c)	0. 0529	0.0446	0.0313	0.0128	0.0089
(d)	0.1103	0.0400	0.0070	0.0034	0.0017

Fig. 5(a)–(d) show the comparison of simulation results for different images of size 256×256. Each column of images includes the original image, the encrypted image, the undecrypted recovered image, the unsubstituted decryption signal and the decrypted recovered image, respectively. It can be seen from the comparison that the recovered image has high similarity with the original image, and the simulation results show that the PSNR of the

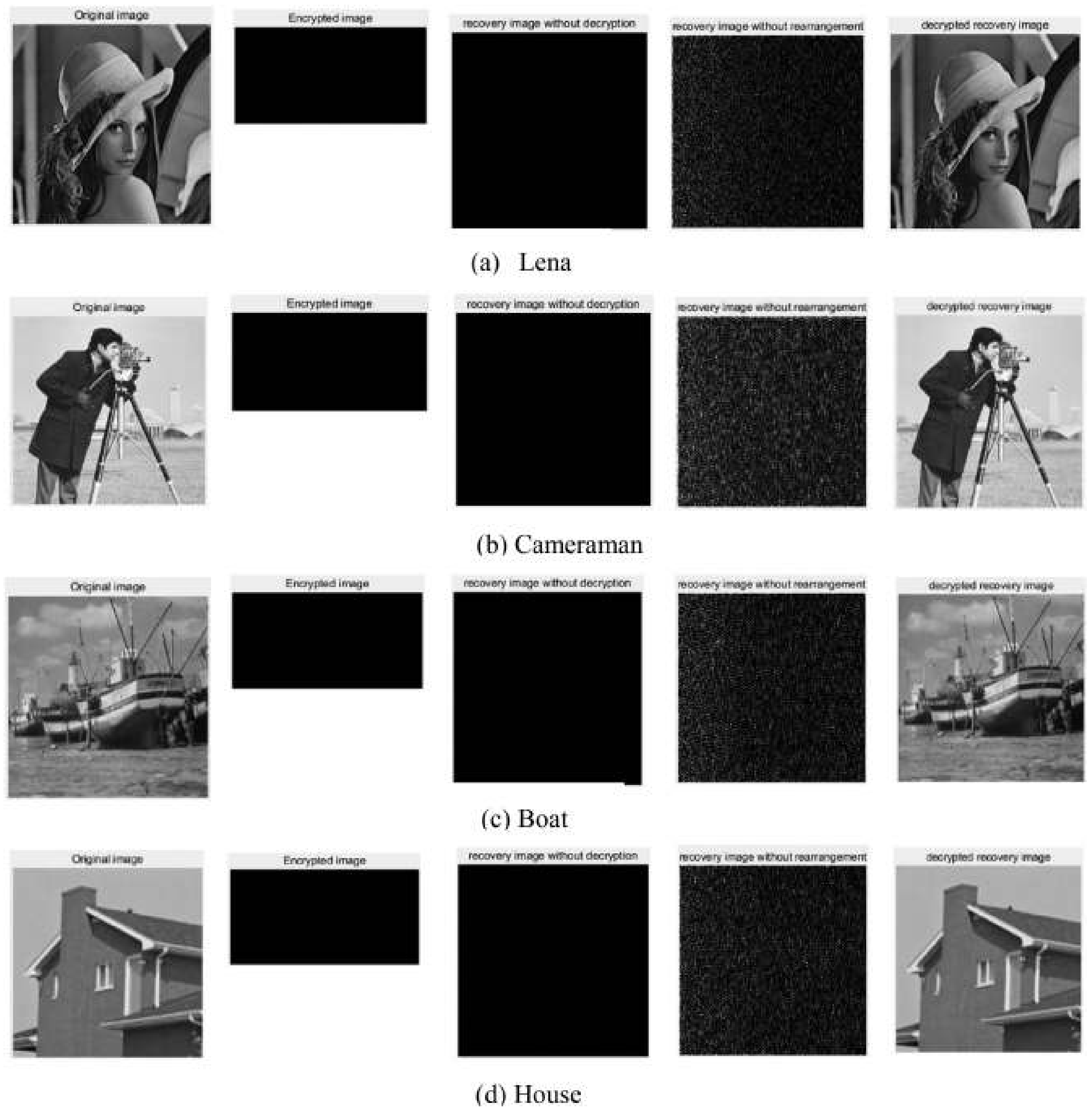

Fig. 5. Simulation comparison of different images.

recovered images after decryption of the above four groups of images exceeds 30db, which indicates that the quality of the recovered images are very high. And it is clear that the encrypted image is completely different from the original image. From the third column, we find that the image without energy decryption cannot display any information of the original image at all. This illustrates that our scheme's energy encryption algorithm has good encryption performance for image signals. Similarly, the unsubstituted decrypted signal also does not display the original image information. Therefore, our scheme can also protect the security of media images.

Fig. 6 shows the PSNR values of the undecrypted, unsubstituted and correctly decrypted of the four images at different compression ratios. From the figure above, the PSNR value of the restored image is very low when no replacement operation is performed. Similarly, if the energy decryption operation is not performed, the original image cannot be restored correctly. Therefore, the replacement operation not only saves storage space but also provides certain security. Moreover, by setting an appropriate threshold, our scheme can achieve a better recovery effect when the compression ratio is 0.3.

To further illustrate the reduced space complexity of using replacement, we present a comparison of the space occupied by high-dimensional information after sparseness in Fig. 7. Here, we set appropriate thresholds to balance the relationship between space and restoration quality. It can be seen from the results that after using the permutation replacement method, the space complexity of the sparse signal matrix decreases by at least two thirds and at most four fifths. Therefore, the use of permutation replacement can greatly reduce the loss of storage space, making our solution available for practical IOT applications.

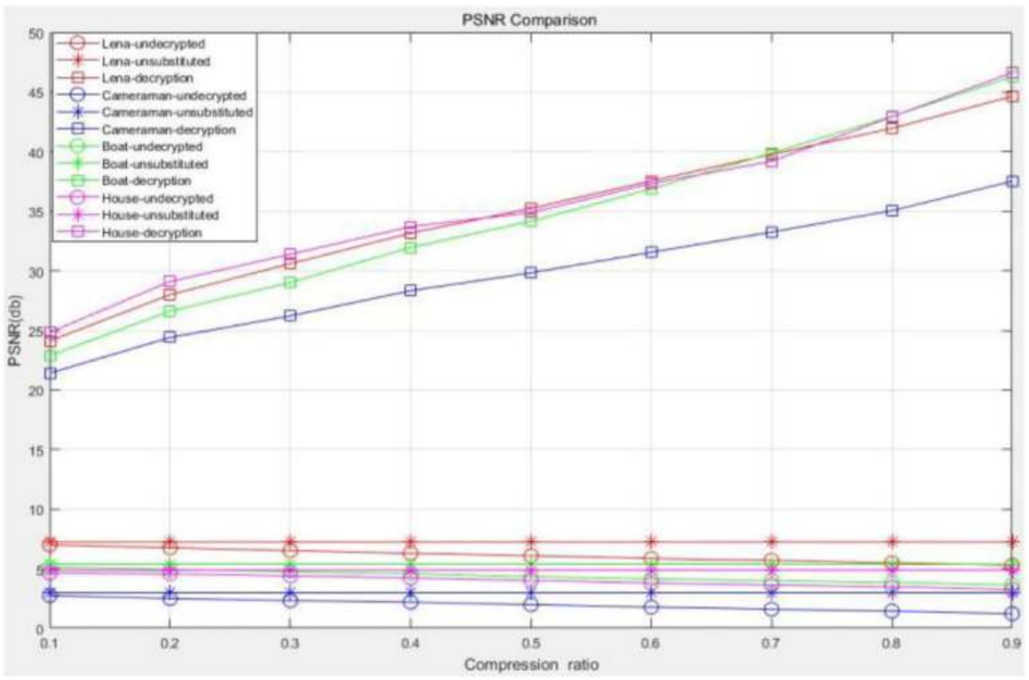

Fig. 6. Comparison of PSNR among Images of undecrypted, unsubstituted and decryption.

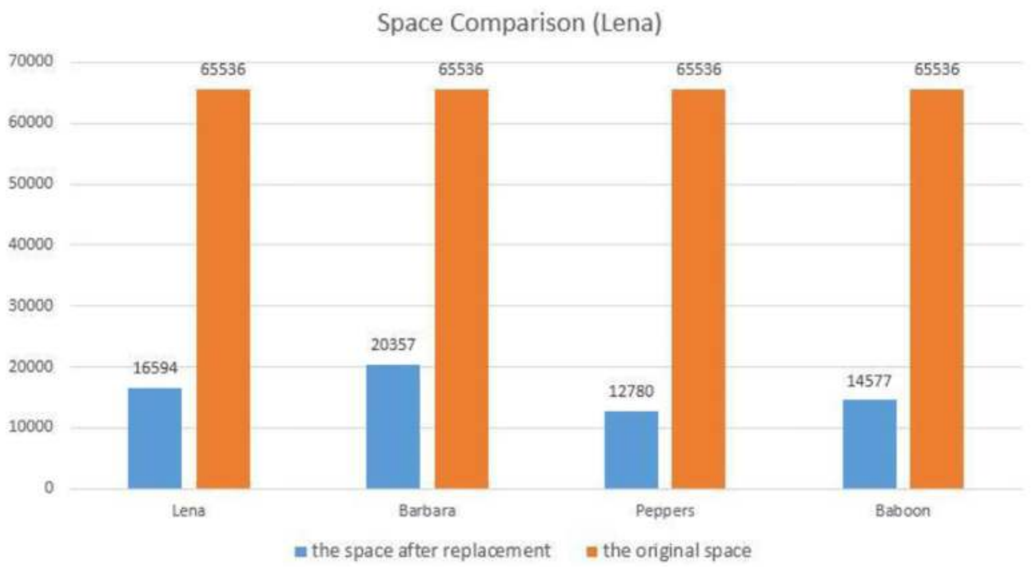

Fig. 7. Space comparison of replacement and non-replacement.

Table 2 shows the PSNR of the recovered image at different compression ratios for different CS encryption schemes. The comparison shows that the recovery effect of our scheme is much better when the compression ratio exceeds 0.3. The quality of the image recovered more than 30db and this means our scheme can achieve a good recovery effect. Compared with other schemes, our scheme maintains good recovery performance with very low compression ratio. So, our scheme can guarantee good recovery performance while ensuring image security.

Table 3 gives a comparison of the client encryption time T_{client} and the server-side decryption time T_{cloud} in the system scheme.

Table 2
PSNR(db) comparison of different CS encryption (CSE) Schemes.

Lena(256 × 256)						
CR	0.2	0.3	0.4	0.5	0.6	0.7
CSE [6,8]	15.63	20.40	24.61	27.51	29.11	30.11
Kryptein [11]	24.81	26.97	29.62	30.76	31.19	33.55
Scheme [12]	28.34	30.43	31.21	31.21	31.21	31.21
Our scheme	28.02	30.60	33.16	35.24	37.54	39.77

Table 3
Time consumption at different compression ratios.

CR Time(s)	0.2	0.3	0.4	0.5	0.6
T_{client}	0.0009	0.0039	0.0037	0.0040	0.0051
T_{cloud}	0.71	1.39	2.82	4.61	7.48

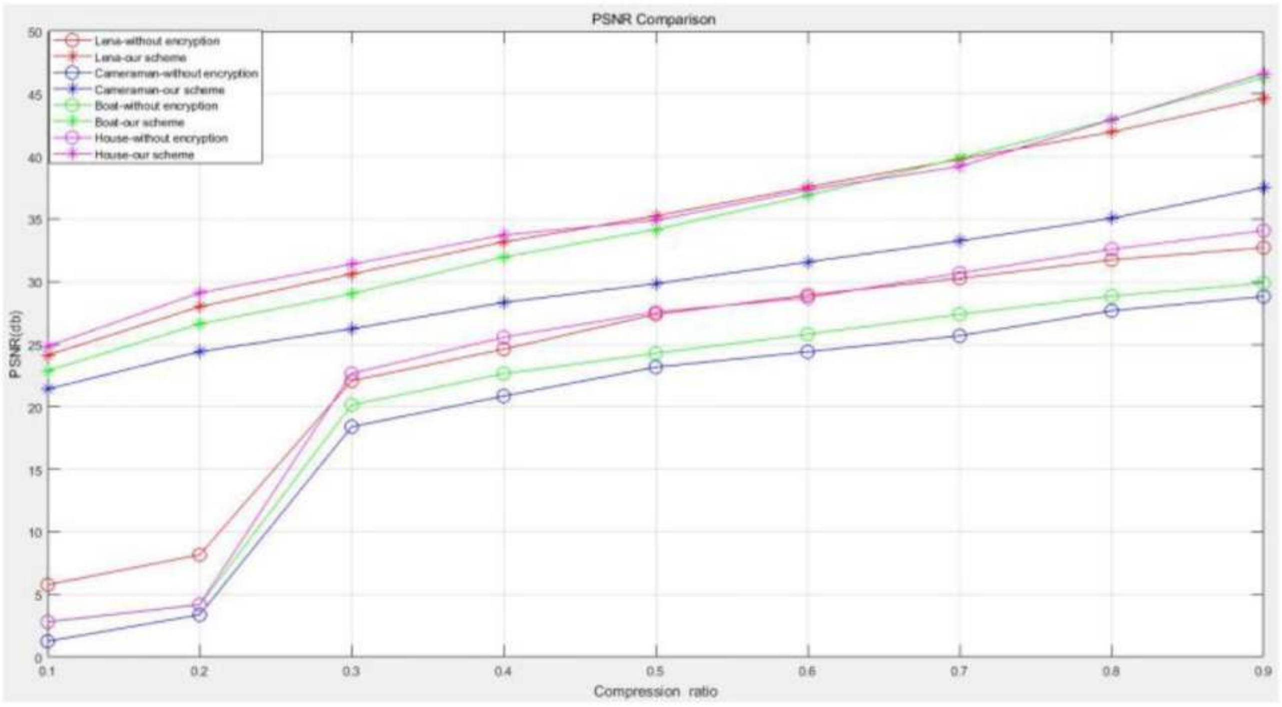

Fig. 8. Comparison of PSNR gain.

Here T_{cloud} includes decryption, error control, and operation to resume the permutation.

Fig. 8 gives the PSNR of the recovered images when lena, cameraman, boat and house four different images are not encrypted and encrypted with our scheme. we can see that the recovery effect of our encryption scheme is better than that of no encryption. And our scheme still maintains certain recovery characteristics in the case of low compression ratios, while the CS method in non-encryption does not recover any image information at the compression ratio below 0.3. In order to clearly demonstrate the advantages of our solution, we provide detailed PSNR gains in Table 4.

Table 4
Comparison of PSNR of the recovered images in different schemes.

Image	CR	No encryption	Our scheme	PSNR gain	Average
Lena	0.3	22.07	30.60	8.53	8.62
	0.4	24.60	33.16	8.56	
	0.5	27.38	35.24	7.86	
	0.6	28.92	37.54	8.62	
	0.7	30.25	39.77	9.52	
Cameraman	0.3	18.40	26.22	7.82	7.34
	0.4	20.85	28.33	7.48	
	0.5	23.16	29.83	6.67	
	0.6	24.39	31.56	7.17	
	0.7	25.67	33.25	7.58	
Boat	0.3	20.15	29.02	8.87	10.33
	0.4	22.64	31.96	9.32	
	0.5	24.27	34.16	9.89	
	0.6	25.79	36.87	11.08	
	0.7	27.38	39.88	12.5	
House	0.3	22.66	31.39	8.73	8.27
	0.4	25.55	33.71	8.16	
	0.5	27.56	34.89	7.33	
	0.6	28.71	37.32	8.61	
	0.7	30.68	39.19	8.51	

6. Conclusion

In this paper, we use CS in the IoT cloud system, and design the methods of identity authentication, information authentication and ciphertext energy encryption to provide a secure solution when faced with the three major security threats in the IoT system. Moreover, we implemented the method of key sharing between users and proved that the method can resist multiple attack methods. In order to make the scheme better applicable to the actual IoT system, we have designed two fault-tolerant models, which can set the appropriate fault-tolerant probability method to make the ciphertext transmission more secure and reliable according to the probability of cloud server failure in the actual IoT cloudy system. Meanwhile, the method of replacing random matrix with replacement vector can save a lot of storage space and maintain high recovery quality. Therefore, our solution can not only improve the security protection of private data in the IoT cloud system, but also reduce the transmission bandwidth and save a lot of storage space. And it has a wide application range under the framework of virtual currency, internet of things and cloud encryption systems.

Acknowledgements

This work is supported in part by the National Key Research and Development Program of China (No. 2016YFB0800601), the Natural Science Foundation of China (Nos. 61303217, 61502372), the Natural Science Foundation of Shaanxi province (Nos. 2013JQ8002, 2014JQ8313).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jisa.2019.04.004.

References

[1] Candès EJ, Romberg J, Tao T. Robust uncertainty principles:exact signal reconstruction from highly incomplete frequency information,". IEEE Trans Inf Theory 2006;52(2):489–509.
[2] Candès EJ, Tao T. Decoding by linear programming,. IEEE Trans Inf Theory 2005;51(12):4203–15.
[3] Donoho DL. Compressed sensing. IEEE Trans on Inf Theory 2006;52(4):1289–1306.
[4] Candès EJ, Tao T. Near-optimal signal recovery from random projections: universal encoding strategies. IEEE Trans Inf Theory 2006;52(12):5406–25.
[5] Rachlin Y, Baron D. The secrecy of compressed sensing measurements,. In: Proceedings of the 46th annual Allerton conference on communication, control, and computing; 2008. p. 813–17.
[6] Zhang M, Kermani MM, Raghunathan A, Jha NK. Energy-efficient and secure sensor data transmission using encompression. In: Proceedings of the international conference on VLSI design & international conference on embedded systems; 2013. p. 31–6.
[7] Gao J, Zhang X, Liang H, Shen XS. Joint encryption and compressed sensing in smart grid data transmission. In: Proceedings of the global communications conference; 2014. p. 662–7.
[8] Pudi V, Chattopadhyay A, Lam K-Y. Secure and lightweight compressive sensing using stream cipher. IEEE Trans Circuits Syst II Express Briefs 2017;65:371–5.
[9] Fay R, Ruland C. Compressive sensing encryption modes and their security. Internet Technol Secur Trans 2017:119–26.
[10] Wang C, Zhangy B, Reny K, Rovedax JM, WenCheny C, Xuy Z. A privacy-aware cloud-assisted healthcare monitoring system via compressive sensing. IEEE INFOCOM; 2014. p. 2130–8.
[11] Hung TH, Hsieh SH, Lu CS. Privacy-preserving data collection and recovery of compressive sensing. In: Proceedings of the IEEE China summit & international conference on signal & information processing; 2015. p. 473–7.
[12] Xue W, Luo C, Lan G, Rana R, Hu W, Seneviratne A. Kryptein: a compressive-sensing-based encryption scheme for the internet of things. In: Proceedings of the ACM/IEEE international conference on information processing in sensor networks; April 2017. (IPSN 2017), 12 pages.
[13] Zhang Y, Zhou J, Zhang LY, Chen F, Lei X. Support-set-assured parallel outsourcing of sparse reconstruction service for compressive sensing in multi–clouds. In: Proceedings of the International symposium on security & privacy in social networks and big data; 2016. p. 1–6.
[14] Zhang Y, Zhou J, Xiang Y, Zhang LY, Chen F. Computation outsourcing meets lossy channel: secure sparse robustness decoding service in multi-clouds.. IEEE Trans Big Data 2017:371–5.
[15] Gubbi J, Buyya R, Marusic S, Palaniswami M. Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener Comput Syst 2013;29(7):1645–60.
[16] Lee I, Lee K. The Internet of Things (IoT): applications, investments, and challenges for enterprises.. Bus Horiz 2015;58(4):431–40.
[17] He D, Chen J, Hu J. An ID-based client authentication with key agreement protocol for mobile client–server environment on ECC with provable security. Inf Fusion 2012;13(3):223–30.
[18] He D, Zeadally S, Kumar N, Lee JH. Anonymous authentication for wireless body area networks with provable security. IEEE Syst J 2017;11:2590–601.
[19] Li W, Li XL, Gao JT, Wang HY. An efficient ID-based mutual authentication and key agreement protocol for mobile multi-server environment without a trusted registration center and ESL attack. In: Proceedings of the Chinese conference on trusted computing & information; 2017. p. 348–69.
[20] Candés E. The restricted isometry property and its implications for compressed sensing. Comptes Rendus Mathematique 2008;346(9):589–92.
[21] Rana R, Yang M, Wark T, Chou CT, Hu W. SimpleTrack: adaptive trajectory compression with deterministic projection matrix for mobile sensor networks. Sens J 2015;15(1):365–73 IEEE 2015.
[22] Pointcheval D, Stern J. Security proofs for signature schemes. In: Proceedings of the advances in cryptology –EUROCRYPT'96, LNCS, 1070. Springer-Verlag Berlin, Heidelberg,; 1996. p. 387–98.
[23] Zhang M, Zhang Y, Jiang Y, Shen J. Obfuscating EVES algorithm and its application in fair electronic transactions in public cloud systems. IEEE Syst J 2019:1–9. doi:10.1109/JSYST.2019.2900723.