Received August 17, 2018, accepted September 20, 2018, date of publication October 1, 2018, date of current version October 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872749

A Secure and Efficient Distributed Storage Scheme SAONT-RS Based on an Improved AONT and Erasure Coding

LILI YAO, JINTIAN LU, JIABING LIU, DEJUN WANG, AND BO MENG
School of Computer Science, South Central University for Nationalities, Wuhan 430074, China

Corresponding author: Bo Meng (mengscuec@gmail.com)

This work was supported in part by the Fundamental Research Funds for the Central Universities, South Central University for Nationalities, under Grant CZZ18003 and Grant QSZ17007, and in part by the Natural Science Foundation of Hubei Province under Grant 2018ADC150.

ABSTRACT Distributed storage has been widely used by enterprises in big data and cloud computing. However, the open nature of distributed storage and the geographical restrictions have constrained distributed storage development. People have put forward higher requirements on the security of node data, especially focusing on confidentiality, recoverability, and integrity. In this paper, we find that there are four security vulnerabilities in AONT-RS and RAONT-RS. In addition, we propose an improved AONT called SAONT in which a canary is not used. After that, we present a secure and efficient distributed storage scheme called SAONT-RS based on SAONT and erasure coding. Finally, the security analysis is given from the four aspects of confidentiality, recoverability, integrity, and anti-short plaintext attack. The experiments show that SAONT-RS has high security and efficiency of node data in distributed storage.

INDEX TERMS Distributed storage, erasure coding, efficiency, information dispersal algorithm.

I. INTRODUCTION

To deal with geographical restrictions and to overcome single point of failure, distributed storage has been introduced [1]. Distributed storage uses the space on each computer in a network and integrates the decentralized space into one to provide virtual storage services. Distributed storage has been widely used by enterprises in big data and cloud computing [2]–[5]. The abilities and skills of attackers are becoming more powerful [6]–[8], for example, foreign giant Dun & Bradstreet's 52GB database was leaked, and Indian McDonald's 2.2 million user data was leaked in 2017 [9], [10]. People have put higher requirements on data security in distributed storage, especially on node data confidentiality, recoverability and integrity [11], [12]. For confidentiality, Mar *et al.* [13] combined secret sharing [14] and an information dispersal algorithm (IDA) [15] to provide storage security for enterprise data. However, it has the disadvantage of being unable to prevent an attack from an adversary that damages node data. Tan *et al.* [16] proposed a threshold-based secret sharing scheme based on the multi-dimensional spherical principle. However, the scheme has limitations on the application because it is only suitable for small data. Xu *et al.* [17] designed a storage model based on threshold public key encryption and exponential erasure coding. The performance was lower because it applies the public cryptosystem to encrypt data and manage the keys. To provide integrity, Hrishikesh and Manjunath [18] presented the AONT-CRS scheme in which a fixed value called a canary is utilized to implement integrity. However, it fails to recover node data correctly when an attack damages node data and may exhaust computing resources in the event of a DoS attack [19] in the reconstruction phase. To address recoverability, Chen *et al.* [20] presented RAONT-RS based on the AONT-RS scheme [21], in which a commitment scheme [22] is applied to correctly recover the stored data. However, the computational efficiency of RAONT-RS is too low to be practical for some special applications.

Therefore, for the purposes of supporting node data security and efficiently encrypting big data, a secure and efficient distributed storage scheme SAONT-RS based on an improved AONT called SAONT and erasure coding is proposed.

The main contributions of this paper are summarized as follows:

①. It studies the important information dispersal algorithms and finds that AONT-RS and RAONT-RS schemes have four existing security vulnerabilities.

②. It proposes an improved AONT called SAONT in which the canary is replaced by an SHA-3 function and the random key is expanded to a 512-bit hash value to strengthen node data security.

③. It presents a secure and efficient distributed storage scheme SAONT-RS that uses an RS erasure code to encode the SAONT package and then to disperse it in the distribution phase to provide recoverability, and it uses an SHA-3 hash function to verify the integrity of node data at the beginning of the reconstruction phase to improve the efficiency of distributed storage.

④. It analyzes the SAONT-RS security from the four aspects of confidentiality, anti-short plaintext attack, recoverability and integrity. Compared to AONT-RS and RAONT-RS, SAONT-RS has the ability to largely prevent short plaintext attacks, correctly and efficiently recover node data at the beginning of the reconstruction phase, and prevents DoS attacks, such as an attack that continually damages the node data in the reconstruction phase.

⑤. It develops an SAONT-RS system to carry out practical experiments. The results show that SAONT-RS achieves high security and efficiency of node data in distributed storage.

The rest of the paper is organized as follows. The related work is introduced in Section II. The AONT-RS and RAONT-RS schemes are reviewed in Section III. An improved AONT called SAONT is presented in Section IV. The SAONT-RS architecture and security analysis are proposed in Section V. Experiments are presented in Section VI, and the conclusion is drawn in Section VII.

II. RELATED WORK

In this section, we describe the state-of-art of distributed storage in detail. Tan *et al.* [16] explained that there are four primary technologies used in data distributed storage methods: multicopy technique, erasure coding, secret sharing scheme, and information dispersal algorithm. Here, we briefly categorize these four technologies into two broad schemes: redundancy-based schemes and confidentiality-based schemes.

A. REDUNDANCY-BASED SCHEME

The redundancy-based scheme provides the reliability of distributed storage systems by adding redundant data.

The redundancy-based scheme includes the multicopy technique and erasure coding [23], [24]. The multicopy technique copies the original data in multicopies and then stores the data in data nodes, which can avoid the failure of a single point to a degree. However, it requires more storage space. Erasure coding splits the original data into k blocks and then uses a complicated encoding algorithm to generate n $(n > k)$ coded blocks. Using at least any k coded blocks of n $(n > k)$ coded blocks can restore the original data. In erasure coding $n - k$ coded blocks are added. In this sense, the $n - k$ coded blocks are redundant data. We think in a degree that erasure coding is a redundancy-based scheme. Erasure coding not only improves storage efficiency but also tolerates the failure of any $n - k$ coded blocks, which greatly improves the reliability of distributed storage system. Hence, most enterprises prefer to use erasure coding to address the failure of a single point in distributed storage system.

B. CONFIDENTIALITY-BASED SCHEME

The confidentiality-based scheme improves data security and reliability through encryption or other complex calculations.

The confidentiality-based scheme includes the secret sharing scheme and information dispersal algorithm. The secret sharing scheme is an important part of cryptography and information security. It is widely used in key management, digital signature, and image processing [25], [26]. The secret sharing scheme [14] is suitable for dispersing confidential data. The idea is the (t, n) threshold mechanism. The secret S is split into n subsecrets, where the length of each secret is the same as S. Any t subsecrets or more can reconstruct the secret S, but any fewer than t subsecrets are unable to obtain any information of the secret S. Nevertheless, the secret sharing scheme has a high storage cost and high computational cost. It is suitable for processing small data or managing the key. The information dispersal algorithm is derived from the IDA algorithm proposed by Rabin [15]; it is also a (t, n) threshold scheme. IDA splits the file into n subfiles and stores them in n different nodes. Any t $(t < n)$ subfiles can reconstruct the original file. IDA is suitable for dispersing big data. At the present, the generalized information dispersal algorithm no longer refers to Rabin's IDA, but a distributed algorithm with the IDA principle. Information dispersal algorithms can be used to implement data security, reliability, fault tolerance, and efficient transmission of information in distributed storage systems. Currently, information dispersal algorithms are widely used in enterprises, for example, Cleversafe and Data Domain, and they are actively researched by IBM and Microsoft [27]–[30].

The information dispersal algorithm is a widely used scheme in distributed storage [31]–[33] and is divided into two categories, unencrypted schemes and encrypted schemes.

1) UNENCRYPTED SCHEME

The idea of the unencrypted scheme as introduced by Rabin does not encrypt the original data, which makes it no need for large storage space and can be used to disperse big data. Compared to secret sharing, each share in Rabin's IDA contains the original information that can be easily attacked by an adversary. Rabin's IDA does not consider privacy security requirements.

2) ENCRYPTED SCHEME

The idea of the encrypted scheme is converting the original data into ciphertext through some kind of encryption mechanism that erases the characteristics of the plaintext and then storing the ciphertext or key. Essentially, it is a (t, n) threshold secret sharing scheme introduced by Shamir's secret sharing scheme [14], which can achieve information theory security, but requires large storage space and high computational demands so that it is only suitable for dispersing small data or providing key security.

The encrypted scheme includes two types, one is Krawczyk's SSMS algorithm [34], which encrypts files with a key-based encryption algorithm first, and then disperses the ciphertext with an IDA and distributes the keys using Shamir's secret sharing scheme to achieve data confidentiality. The SSMS algorithm can achieve theoretical security and solve the problem of large storage space overhead caused by Shamir's secret sharing. However, the computational demand is still not small. To encrypt a file with a cryptosystem not only leads to high computational demands but also requires key management. In 2016, Tan *et al.* [16] proposed a threshold-based secret sharing scheme based on the multidimensional spherical principle, which transforms the original secret into an m-sphere central coordinates and then into n shadow secret coordinates distributed to n participants and then takes k linear uncorrelated coordinates to determine the unique sphere center and recover the secret to realize the security of confidential information in cloud storage. Xu *et al.* [17] designed a storage model based on threshold public key encryption and exponential erasure coding to satisfy the requirements of confidentiality and fault tolerance in cloud storage. This storage model includes clients, storage servers, and key servers. The client generates a session key for the stored data and encrypts the stored data using the session key, and then the storage servers encode the encryption data with an exponential erasure code. Finally, it deals with key management using secret sharing scheme.

The other is the AONT algorithm, presented by Rivest [35] in 1997, which is used to preprocess the original data. It can encrypt big data with no need to manage the key. In 2011, Resch and Plank [21] proposed AONT-RS to deal with two vulnerabilities, which include Shamir's secret sharing scheme failure in dispersing large files and that Rabin's IDA [15] does not provide data security. AONT-RS can not only disperse large files but also satisfies computational security in distributed storage systems. In 2013, Abreha and Shetty [36] analyzed the AONT algorithm and determined it could solve security, and the erasure code addresses fault tolerance. In 2014, Hrishikesh and Manjunath [18] presented AONT-CRS which combined Resch's AONT algorithm and the CRS erasure code to provide security and performance. In 2016, Mar *et al.* [13] combined secret sharing and IDA with more flexible authentication and access control-based security to provide storage security for enterprise cloud data. The scheme used a lightweight AONT algorithm to encode the files and disperse them into different nodes using IDA. The metadata server stores the metadata which includes crucial information, and then clients utilize crucial information to store, retrieve, and reconstruct files. To prevent metadata leakage, it uses a secret sharing algorithm to ensure the security of crucial information. In 2017, Chen *et al.* [20] discussed AONT-RS and presented RAONT-RS. They explained that information leakage may occur if the plaintext is too short. In addition, RAONT-RS used a commitment scheme [22] to make the AONT-RS scheme more robust. They claimed that it is more robust and secure than AONT-RS.

The studies [37]–[40] showed that Rabin's IDA can disperse big data, but the attacker can easily extract the information from these files. Shamir's secret sharing scheme can provide perfect security, but it is suitable only for small data.

III. THE REVIEW OF AONT-RS AND RAONT-RS

In this section, we analyze the classical AONT-RS scheme [21] and RAONT-RS scheme [20], find that there are four security vulnerabilities:

1) Correct recovery is not efficient, so it could be limited in adoption due to its computational demands.
2) It is largely unable to prevent short plaintext attacks.
3) It fails to prevent DoS attacks, such as an attack that continually damages the node data in the reconstruction phase.
4) It cannot correctly and efficiently recover the node data that was damaged by the attacker.

A. AONT-RS

AONT-RS was the first scheme to combine a variant of Rivest's AONT algorithm and the Reed-Solomon (RS) erasure code [41] and achieve security and good performance for big data in distributed storage systems. AONT-RS assumes that there is a symmetric cipher E with a cipher block chaining (CBC) mode, a hash function SHA-256, a random key k and a systematic erasure code RS. The process of AONT-RS is shown in Fig. 1. The stored data are encrypted by E and output ciphertext C; ciphertext C is accepted as an input to hash function SHA-256 to generate a hash value that has the same length as the random key k so that key k can enable the bitwise exclusive-or (XOR), the hash value to produce a value b. Additionally, the scheme uses a canary, a known, fixed value to verify integrity. After decoding the AONT package, the canary can verify whether the node data have been damaged. Finally, the ciphertext C (includes the encrypted canary) and value b make up the AONT package.

However, AONT-RS has some security vulnerabilities. The first vulnerability is information leakage. The hash function SHA-256 is a version of SHA-2, which means the length of the hash value is 256 bits. When the stored data length is too short, each share is shorter. Even though the attacker does not obtain any k shares, he may obtain the stored data by attacking the hash value to cause information leakage. The second vulnerability is the inability to correctly recover the node data, and it cannot prevent a DoS attack, such as an attack that continually damages the node data. AONT-RS uses a fixed value canary to verify the integrity of the node data in decoding the AONT package. Although this method can ensure data integrity, it is unable to correctly recover the node data because the process of reconstruction ends after

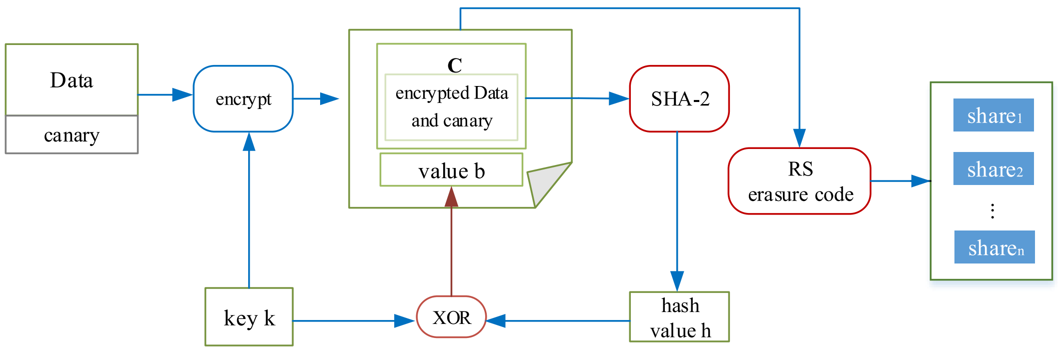


FIGURE 1. The process of AONT-RS.

finding the damaged data. In particular, if the attacker uses this vulnerability to continually damage the node data in the reconstruction phase to launch a DoS attack, this will cause the user's machine to crash.

B. RAONT-RS

In 2017, Chen *et al.* [20] proposed RAONT-RS which is an improvement of the AONT-RS scheme.

First, it analyzed two cases that may cause information leakage. A simple example of AONT-RS with the (4, 5) threshold is shown in Fig. 2. In AONT-RS, the stored data is processed with the AONT algorithm and then receives an AONT package that includes two parts: C and C_d. C refers to the entire ciphertext of the stored data, and C_d refers to the value b calculated by the key k and the hash value. It assumes that C is 128 bits as long as key k; V_1 and V_2 represents the whole C; and V_3 and V_4 represents the whole C_d. After using RS erasure coding, the stored data is coded into five shares, the first four shares store V_1, V_2, V_3 and V_4. The fifth share is a redundant block. One case of information leakage is that if an attacker knows all of C and a part of C_d; in this case, if an attacker obtains three shares at random, the attacker will obtain the whole C and half of C_d, so he can determine exactly half of the k, which is information leakage. The other case is if the attacker knows all of C_d and all but any single bit of C since the missing bit is either 0 or 1, there are two possibilities for guessing C. After obtaining C, the attacker uses SHA-2 to calculate the hash value either h_0 or h_1. In addition, according to $h \oplus b = k$, the attacker knows two kinds of keys. Lastly, the attacker uses the two keys to decrypt the stored data, either D_1 or D_2 that causes information leakage. RAONT-RS uses the method of specifying the length of ciphertext C with some conditions to address information leakage in these two cases.

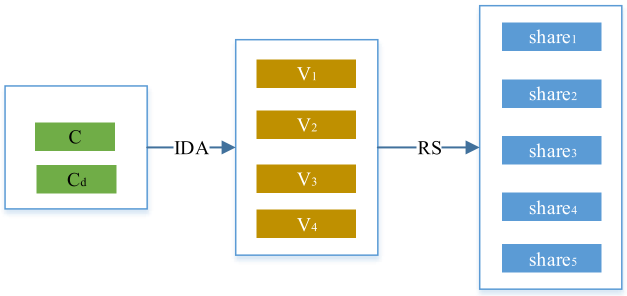


FIGURE 2. An example of AONT-RS with (4, 5) threshold.

Second, it uses a commitment scheme to recover the node data correctly. The commitment scheme compares each bit to verify the integrity of the ciphertext C and to recover the correct data. RAONT-RS uses IDA to split C into multiple shares and employs commitment scheme to calculate each bit of a share. When decoding the AONT package, RAONT-RS uses the calculated values to verify whether each bit of a share is true. If one bit is wrong, the share can be recovered; this loop ends at the end of C.

Chen *et al.* [20] proposed RAONT-RS and discussed AONT-RS and found that there are two vulnerabilities. First, there is information leakage in two cases. One case is when the attacker knows all of C and a part of C_d and the attacker may know half of the key k. The other case is if the attacker knows all of C_d and all of C except for any single bit of C, then the attacker knows the stored data. Second, RAONT-RS removes the canary and employs the commitment scheme to recover the node data.

However, RAONT-RS does not completely address the vulnerabilities of AONT-RS and also produces new problems. First, RAONT-RS specifies the length range of the ciphertext C, which makes it unable to prevent short plaintext attacks and fails to resist hash function attacks because RAONT-RS uses the SHA-256 hash function in which the length of key k is a maximum of 256 bits. Next, RAONT-RS employs the commitment scheme to recover the node data, but the computational efficiency of the solution is too low to be practical for some applications. In addition, when experiencing a DoS attack in AONT-RS, such as an attack that continually damages the node data in the reconstruction phase, RAONT-RS cannot prevent it. Based on the analysis of AONT-RS and RAONT-RS, we conclude that there are still four security vulnerabilities:

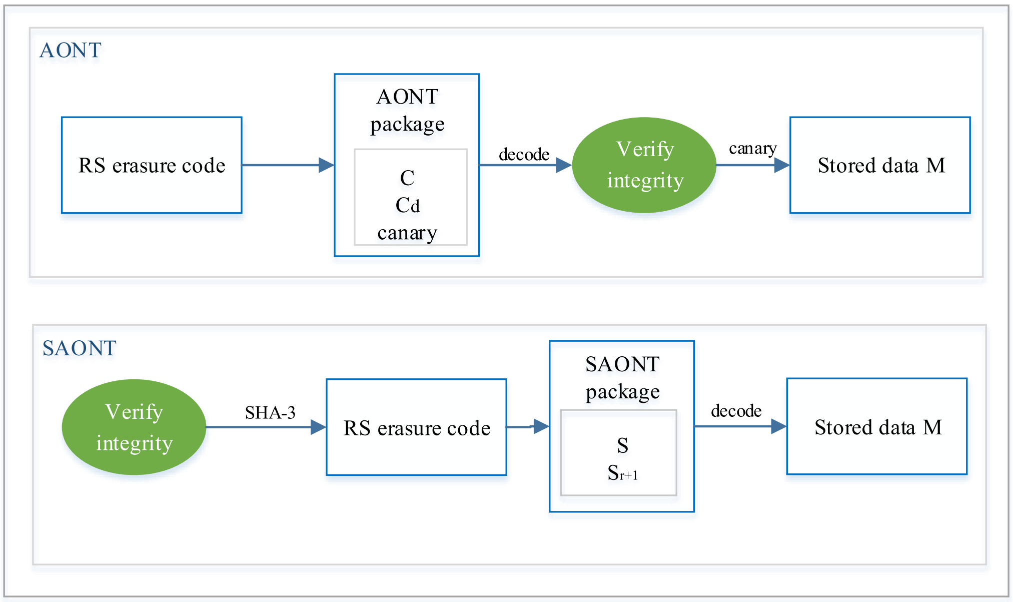


FIGURE 3. Comparison between SAONT and AONT in verifying integrity and recovering node data.

1) Correct recovery is not efficient, so it could be limited in adoption due to its computational demands
2) It cannot correctly and efficiently recover the node data which damaged by the attacker.
3) It fails to prevent Dos attacks, such as an attack that continually damages the node data in the reconstruction phase.
4) It is largely unable to prevent short plaintext attacks.

IV. AN IMPROVED AONT ALGORITHM CALLED SAONT

In the review of the AONT-RS and RAONT-RS schemes in section III, we found that AONT-RS and RAONT-RS have four security vulnerabilities. In this section, to address the four security vulnerabilities, we propose an improved AONT called SAONT in which the canary is replaced with an SHA-3 hash function and the random key is expanded to a 512-bit hash value to enhance the security of the node data.

SAONT processes the stored data M to produce the SAONT package composed of a ciphertext S and a cipher block S_{r+1}. The ciphertext S is generated by a symmetric cipher, and the cipher block S_{r+1} is calculated by the hash value of S and a random key K_A, and then the SAONT package is dispersed using an RS erasure code. The SAONT algorithm is different from AONT-RS and RAONT-RS in verifying integrity and recovering node data, as shown in Fig. 3. AONT-RS and RAONT-RS verify the integrity of node data after the AONT package is decoded. However, SAONT-RS verifies the integrity at the beginning of the reconstruction phase, so not only the damaged data has a chance to be correctly recovered, but it also prevents attacks that continually damage the stored data M. In addition, SAONT uses the hash function SHA-3 to produce a 512-bit hash value of the ciphertext and to expand the random key to 512 bits so that it largely prevents the short plaintext attack. In addition, using SHA-3 to verify the integrity of node data at the beginning of the reconstruction phase can not only detect the damaged data but also prevent an attack that continually damages the node data in the reconstruction phase. Finally, the hash values and the RS erasure code are compared to correctly recover the damaged data that resulted from an attack launched by an adversary.

The differences among SAONT-RS, AONT-RS and RAONT-RS is that SAONT-RS uses SHA-3 and an extended random key.

A. SHA-3

The secure hash algorithm (SHA-1) is designed to have a similar structure and basic mathematical operations as MD5 and SHA-0, both of which have been broken. In 2017, Google announced on the Shattered.it website that it had broken SHA-1 [42]; accordingly SHA-1 was considered insecure and was gradually replaced by SHA-2. However, SHA-2 is similar to its predecessor in design and basic mathematical operations. Shortly thereafter, the SHA-2 defects were discovered and replaced. Subsequently, in 2007, NIST announced a competition to produce the next generation NIST hash function standard, and then NIST announced the winner of the SHA-3 design in 2012 [43]. Afterwards, SHA-3 became a national standard as the latest standard of the hash function in 2015 [44]. The SAONT algorithm uses SHA-3 in two ways. First, SHA-3 calculates a 512-bit hash value of the ciphertext so that it extends the length of the hash value to increase the computation to enhance data security and largely prevent short plaintext attacks. Second, it calculates

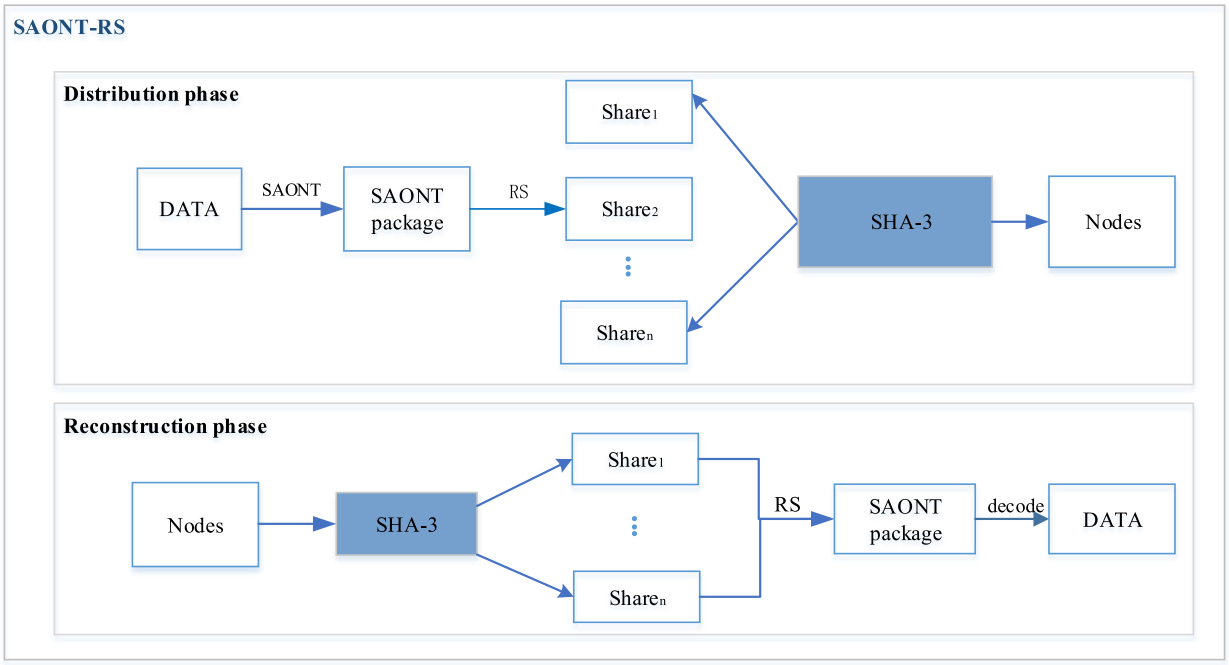


FIGURE 4. Secure and efficient distributed storage scheme SAONT-RS architecture.

the 512-bit hash values of shares in nodes and compares the hash values of the shares at the beginning of the reconstruction phase to verify the integrity of the node data.

B. THE EXPANDED RANDOM KEY

The SAONT algorithm applies the symmetric cipher AES and a random key K_A to encrypt the stored data. Due to the maximum 256 bits length of the random key, the XOR operation between the hash value of S and K_A cannot be calculated directly. To solve this problem, K_A is expanded to 512 bits by padding a one followed by a number of zeros to satisfy different security strength requirements. Before encrypting the random key K_A using a stream cipher, K_A is expanded to the same length as the hash value of S. SAONT expands the random key to enhance the security of the node data and provides confidentiality so that an attacker cannot obtain any information, even if he obtains single or multiple nodes' data less than the threshold k.

V. THE ARCHITECTURE OF SECURE AND EFFICIENT DISTRIBUTED STORAGE SCHEME SAONT-RS

In this section, we introduce the secure efficient distributed storage scheme SAONT-RS architecture in detail. In addition, we analyze the security from four aspects. SAONT-RS is composed of a distribution phase and a reconstruction phase. SAONT-RS applies the SAONT algorithm and RS erasure code to implement the security, recoverability and efficiency of node data. It uses the SAONT algorithm to process the stored data to achieve node data confidentiality and then combines hash function SHA-3 with the RS erasure code to ensure recoverability and efficiency in distributed storage. Fig. 4 depicts the SAONT-RS architecture.

In the distribution phase, SAONT-RS generates a SAONT package. In addition, a SAONT package is distributed to different nodes in a distributed network. In the reconstruction phase, SAONT-RS reconstructs the SAONT package to recover the correct stored data M.

A. DISTRIBUTION PHASE

The distribution phase includes two processes for coding the SAONT package and splitting the SAONT package.

1) CODING THE SAONT PACKAGE

Coding the SAONT package to encode the stored data using the SAONT algorithm. The details are shown in Fig. 5. The stored data M is divided into r words $(M_1 \ldots M_i \ldots M_r)$ using the CBC mode, where M_i is a b bits word. M_i is used to generate coded block S_i by the formula (1).

$$S_i = M_i \oplus E\left(K_A, i\right); \tag{1}$$

where K_A is a randomly generated key, E is a key-based encryption algorithm, e.g., AES-256, i is a variable starting from 1, M_i is a plaintext block, and S_i is a coded block.

The procedure for coding the SAONT package is as follows:

(1) The stored data M is divided into r words $(M_1 \ldots M_i \ldots M_r)$. Each word has a length of b bits, and AES is selected as an encryption algorithm. The random key K_A is used as a key for AES encryption. Variable i is a random plaintext of AES. Then M_i plaintext blocks are calculated

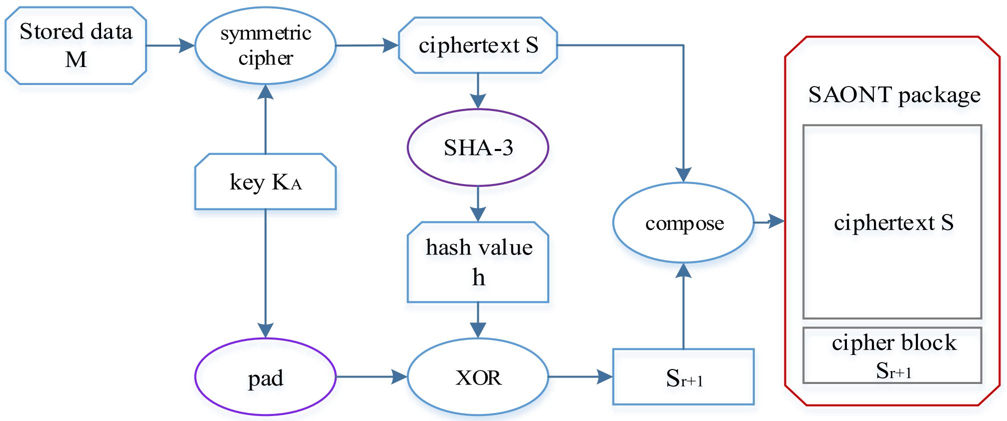


FIGURE 5. Coding the SAONT package.

by formula (1) to obtain r coded blocks($S_1, \ldots, S_r$), i.e., ciphertext S.

(2) SHA-3 is used to calculate the hash value h of the ciphertext S to generate a 512-bit h. Then h is taken as the key of the stream cipher, K_A is input to the stream cipher, and then K_A is XORed with hash value h. However, the random key K_A in AES encryption is a maximum of 256 bits. To perform the XOR operation successfully, K_A is expanded to 512 bits with a one followed by a number of zeros. Finally, it produces a cipher block S_{r+1}, as shown in Fig. 6.

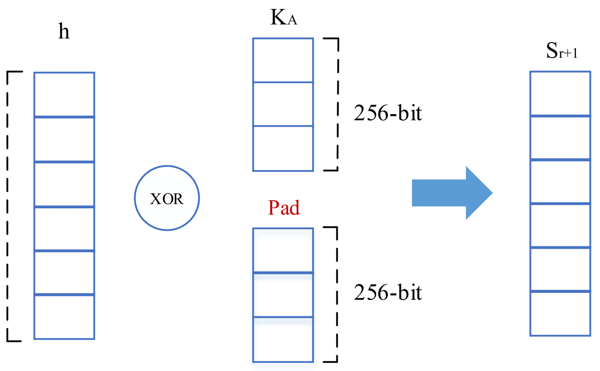


FIGURE 6. Expanding the key K_A to 512 bits.

(3) ($S_1 \ldots S_{r+1}$) is a SAONT package.

2) SPLITTING THE SAONT PACKAGE

Splitting the SAONT package encodes the SAONT package and then disperses it with an RS erasure code. The SAONT package is encoded and dispersed into n shares with the RS erasure code. At the same time, the hash values of the shares are calculated with the SHA-3 function. Finally, the shares are stored in different nodes and the hash values of the shares are stored in the central node.

The procedure for splitting the SAONT package is as follows:

(1) The coding algorithm of the RS erasure code encodes and then disperses the SAONT package. The process is as follows. First, it takes the SAONT package as the input, and splits it into k data blocks to obtain get a k rows of a column in matrix A. Second, there is a generation matrix G of n rows and k columns, the first k rows of which consist of a unit matrix, and the last $n - k$ rows of which are composed of a Vandermonde matrix or Cauchy matrix. Next, the generator matrix G multiplies the matrix A and obtains n shares (y_1, ..., y_n). Finally, since the generation matrix G contains a unit matrix, the y_1, ..., y_k shares are equal to the data blocks A_1, ..., A_k, thus the n shares contain $n - k$ redundant shares and k data shares.

(2) Calculates the hash value of each share y_i using hash function SHA-3 to obtain the hash values $h_1 \ldots h_n$.

(3) Store the hash values of the shares in the central node for safekeeping, and stores the shares in different nodes. Finally, complete the entire process of splitting the SAONT package. The entire process is shown in Fig. 7.

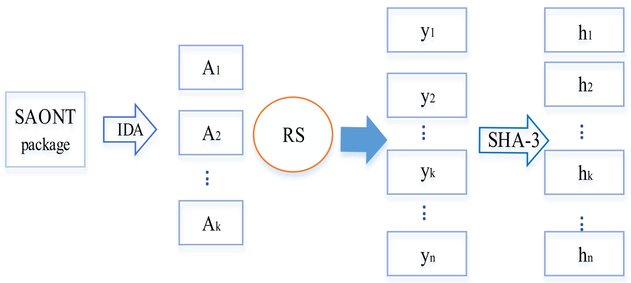


FIGURE 7. Splitting the SAONT package.

B. RECONSTRUCTION PHASE

The reconstruction phase includes processes of reconstructing the SAONT package and decoding the SAONT package.

1) RECONSTRUCTING THE SAONT PACKAGE

Reconstructing the SAONT package is integrates the shares in the storage nodes. Fig. 8 presents the process of reconstructing the SAONT package.

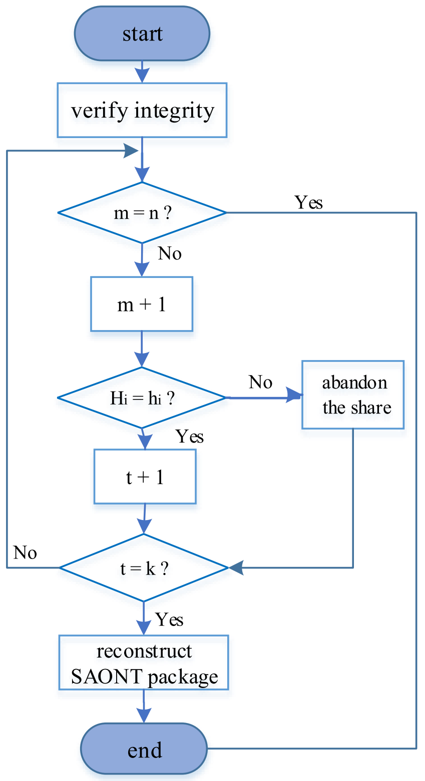


FIGURE 8. Reconstructing the SAONT package.

The procedure of reconstructing the SAONT package is as follows:

(1) Determine whether the number of obtained shares m equals the number of shares n. If yes, ends the process of reconstructing the SAONT package; if not, obtain the share from the storage nodes and calculate the hash value H_i of the share, then the number of m adds one and goes on to the next step.

(2) Compare H_i with h_i to verify the integrity of each share. If H_i equals h_i, the number of reconstructed shares t adds one, and we continue judging whether number t satisfies the threshold k. If H_i is not the same as h_i, this indicates that the share may be damaged, so we discard the damaged share and judge whether number t satisfies the threshold k.

(3) If number t satisfies threshold k, we can reconstruct the SAONT package using the RS erasure code; if not, we continue verifying the integrity of the next share from the first step.

(4) Employ the RS erasure code to recover the correct SAONT package at the beginning of the reconstruction phase, and then end the process of reconstructing the SAONT package.

2) DECODING THE SAONT PACKAGE

Decoding the SAONT package is the reverse process of encoding the AONT package process. After the data are reconstructed, the correct SAONT package is obtained, which contains the ciphertext S and the cipher block S_{r+1}.

The process of decoding the SAONT package is shown in Fig. 9.

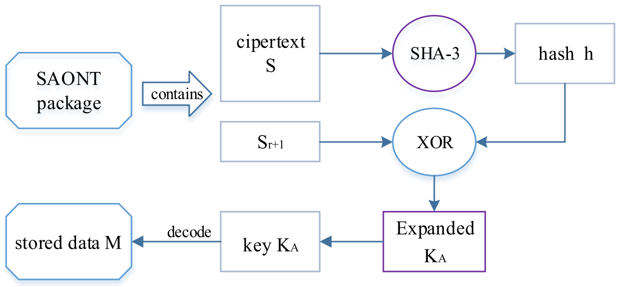


FIGURE 9. Decoding the SAONT package.

(1) The hash value h of the ciphertext S is calculated using hash function SHA-3.

(2) The hash value h is XORed with the cipher block S_{r+1} based on the decryption of the stream cipher to obtain the expanded key; at the same time, the 256 padded bits are removed from the expanded key to obtain the random key K_A.

(3) The stored data M is obtained with the random key K_A.

C. SECURITY ANLYSIS

SAONT-RS uses the SAONT algorithm and RS erasure code to deal with the four security vulnerabilities in AONT-RS and RAONT-RS and to achieve confidentiality, anti-short plaintext attack, recoverability and integrity of node data. We analyze the four security aspects of SAONT-RS.

1) CONFIDENTIALITY

SAONT-RS has the confidentiality so that the adversary cannot decrypt any information when the number of compromised nodes is less than the threshold k.

In the encoding process of the distribution phase, the stored data is encrypted applying a symmetric cipher to generate the ciphertext S, and then the SHA-3 is used to calculate a 512-bit hash value h of the ciphertext S to enhance the confidentiality of the node data. At the same time, to improve the confidentiality of the node data, the random key K_A is expanded with the same length as hash value h to satisfy different security strength requirements. Lastly, h is XORed with the extended key K_A to generate a cipher block S_{r+1} which is the ciphertext of K_A, so that we do not need to safely keep key K_A, and it is not easy for the attacker to obtain the key. Fig. 10 presents the reason for this. When an attacker wants to learn the key K_A, he must obtain the ciphertext S and S_{r+1} first, but this is also very difficult for the attacker. After the SAONT package is generated, it is encoded and dispersed to n different nodes with an RS erasure code. If an adversary wants to obtain some information about the node

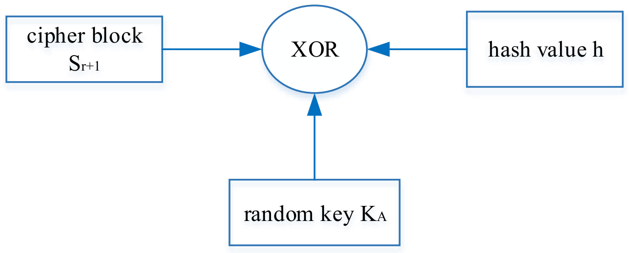


FIGURE 10. Obtaining key K_A.

data, he must obtain more than k available shares at the beginning of the reconstruction phase. Otherwise, the adversary cannot decrypt any information. Thus, K_A and the k shares are the critical information in SAONT-RS to ensure data confidentiality. Compared to the AONT-RS and RAONT-RS schemes, SAONT-RS uses SHA-3 to calculate a 512-bit hash value h and expand the random key K_A to 512 bits to enhance the security of the node data.

2) ANTI-SHORT PLAINTEXT ATTACK

SAONT-RS applies the latest hash function SHA-3 to largely prevent short plaintext attacks.

When the stored data is shorter, the shares are shorter. If the share is too short, the computational cost of a brute-force search will be greatly reduced, and the calculation time will decrease. For short plaintext attacks, if the hash value is not long enough, the adversary can attack the plaintext directly without knowing k or more pieces of the shares. We know that AONT-RS and RAONT-RS largely cannot prevent the short plaintext attack. By contrast, with AONT-RS and RAONT-RS, SAONT-RS applies the latest hash function SHA-3 and expands the key K_A and uses the 512-bit hash value h and the 512-bit expanded key K_A; thus, the security problem caused by short plaintext is addressed to a great extent.

3) RECOVERABILITY

SAONT-RS can correctly and efficiently recover node data using an RS erasure code and hash function SHA-3 at the beginning of the reconstruction phase.

AONT-RS can provide the availability of distributed storage with node failure due to physical damage. AONT-RS uses a fixed value canary that can verify whether the stored data is damaged. However, it is unable to recover the damaged data in nodes. Compared to AONT-RS, SAONT-RS applies the RS erasure code and the hash value of the shares to correctly recover the node data. First, it compares the hash values of the shares so that the damaged data can be found in the reconstruction phase, and then we will abandon the damaged node data (less than $n - k$) so that the remaining shares are not damaged. At the same time, it uses the RS erasure code principle to correctly recover the node data to obtain the correct SAONT package. Finally, it uses SAONT to decode the correct plaintext.

RAONT-RS employs a commitment scheme and can correctly recover the stored data. However, the computational efficiency is too low to be practical for some applications. As an example, there is a stored data M, a (5, 9) threshold RS erasure code. The stored data M is dispersed into 9 shares V[i] by RS. The computational demand of RAONT-RS during recovery operations requires computing 9 decommittal data R[i] and 9 committal data H[i] via a commitment algorithm, and each committal data is dispersed into 9 fragments S[i] via a (5, 9) threshold RS erasure code [20]. However, the computational demand of SAONT-RS during recovery operations requires computing only 9 hash values h[i]. Compared to RAONT-RS, the computational demand of SAONT-RS is obviously lower than RAONT-RS.

4) INTEGRITY

Hash function SHA-3 is used to verify the integrity of the node data at the beginning of the reconstruction phase and to prevent DoS attacks, such as an attack that continually damages the node data in the reconstruction phase, and improve the efficiency of distributed storage.

AONT-RS uses the CBC mode to add a fixed value called a canary to check the integrity of the stored data after decoding the AONT package. However, we found that the efficiency is low and it fails to prevent DoS attacks due to the verify integrity step at the last step in decoding the AONT package process. The attacker can use this vulnerability to launch a DoS attack that constantly damages the shares in the data nodes while reconstructing the AONT package process in the reconstruction phase, which can exhaust the user's computer resources. RAONT-RS employs commitment scheme to verify the integrity of the node data by calculating each bit of a share, which is not only unable to prevent Dos attacks but also has low efficiency.

To address this vulnerability, SAONT-RS uses the hash function SHA-3 to verify the integrity of the shares in the nodes at the beginning of the reconstruction phase and prevents the DoS attack and improves the efficiency of SAONT-RS in the beginning. This is because, if we find that the damaged shares are less than the threshold k, SAONT-RS will end the useless steps so that the attacker has no chance to launch a DoS attack to make the user's computer crash; thus, improving the efficiency of distributed storage.

VI. EXPERIMENTS

In this section, we use the C ++ language to develop the SAONT-RS system running on the Windows 10 Professional 64-bit operating system. In our tests, text data and picture data with different sizes are accepted as the input to the SAONT-RS system to test the efficiency and storage of the ciphertext S and the cipher block $S_{r+.1}$ in the SAONT package.

A. EFFICIENCY

Here, we consider two scenarios for efficiency. One is that there is no attack damaging the node data. The other is that there exists an attack damaging the node data. For the scenario with no attack, we consider two factors on efficiency. One factor is the data type, for example, text or picture. The other is the data size. The test results include the coding time of random key generation and expansion, the AES algorithm and SHA-3. Fig. 11 and Fig. 12 present the test results of the encrypted text data and the encrypted picture data with different sizes from 4KB to100KB.

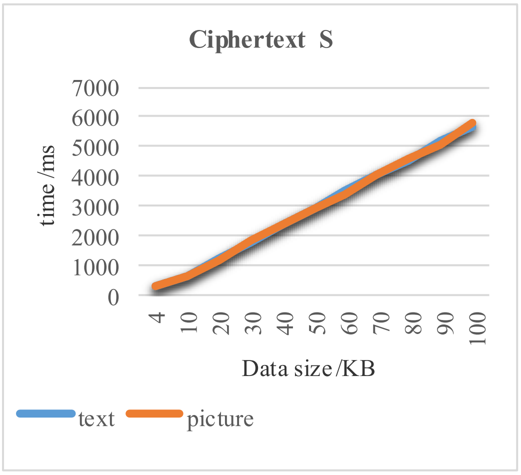


FIGURE 11. The encoding time of the ciphertext S of text and picture.

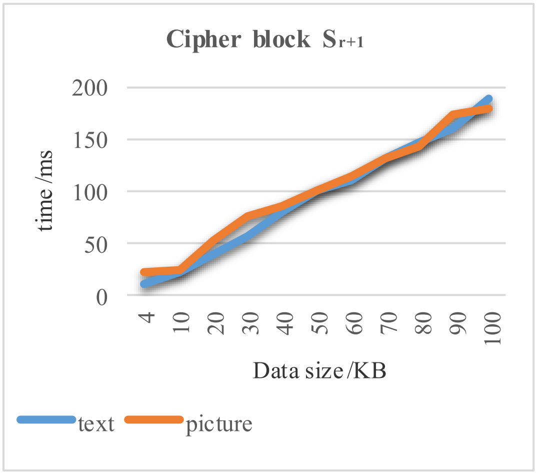


FIGURE 12. The encoding time of the cipher block S_{r+1} of text and picture.

First, we test the relationship between the data types and efficiency. Compared to the encoding time of ciphertext S with text data, the encoding time of ciphertext S with picture data is approximately equal. Compared to the encoding time of the cipher block S_{r+1} with text data, the encoding time of the ciphertext S_{r+1} with picture data is approximately the same. The results indicate that the encoding time is not closely related to the data types, specifically, text data and picture data.

Then, we consider the relationship between data size and efficiency. The results show that the encoding time is proportional to the data size with different data types, specifically, text or picture. Fig. 11 and Fig. 12 show that as the size of the data increases, the time required for encoding also increases. After that, to express the results in detail, a test was implemented using OpenSSL 0.9.8k with a block size of 8 KB in [21], the results in Table 1 show that the coding rate of AES-256 and SHA-2(256) is lower than the coding rate of RC4-128 and MD5. To encode w bytes of data using AONT, both the cryptographic and hash function must process w bytes. Accordingly, the efficiency of SAONT-RS is lower than AONT-RS because SAONT-RS uses SHA-3 twice in the distribution phase and the reconstruction phase, while AONT-RS uses SHA-2 once. In addition, the threshold choice of the RS erasure code also affects the efficiency of SAONT-RS, the larger the threshold value, the more shares, the more calculations of the hash values of the shares.

TABLE 1. The coding rates of cryptographic and hash function.

Function	Coding rate（MB/S）
RC4-128	414.17
AES-256	143.30
MD5	559.47
SHA-2(256)	160.03

For the attack scenario, the efficiency of SAONT-RS is higher than AONT-RS and RAONT-RS. If the stored data is damaged, it means that the AONT package reconstructed by RS has the wrong information. In AONT-RS scheme, only the canary is used after decoding the AONT package to verify whether the stored data is correct. Once the attacker uses this vulnerability to launch an attack that continually damages the shares in the data nodes in the reconstruction phase, the reconstructed package cannot be found incorrect until the end of each decoding AONT package. Then, AONT-RS reconstructs the stored data uses another shares till the stored data are completely reconstructed. However, it not only causes the AONT-RS system crash but also wastes huge computing resources. In RAONT-RS scheme, it is need more computational demands to correctly recover the node data and to address a DoS attack. SAONT-RS verifies integrity of shares before reconstructing the node data with RS, it has the ability to address the vulnerability using hash function SHA-3 at the beginning of reconstruction phase.

B. STORAGE

Taking a 4 KB data block as input data to SAONT with the threshold (10, 16) RS erasure code to test the storage of SAONT-RS system.

Using CBC mode divides the 4 KB data block into 256 slices. A slice is composed of 16 bytes. First, the data block is processed based on formula (1) of the SAONT algorithm to generate the 4096-byte ciphertext S. Then, we calculate the 64-byte hash value of ciphertext S using the hash function SHA-3. After that, we expand a 32-byte random key

TABLE 2. Storage space of SAONT-RS, RAONT-RS and AONT-RS.

SAONT-RS		RAONT-RS		AONT-RS	
Key KA	32bytes	Key k	32bytes	Key k	16bytes
Hash value	64bytes	Hash value	32bytes	Hash value	16bytes
S	4096bytes	C	4096bytes	C	4096 bytes
Expanded key	64 bytes	Value b	32bytes	canary	16 bytes
S_{r+1}	64 bytes	Expand data	2 bytes	Value b	16 bytes
Expand data	0 bytes	Expanded key	0 bytes	Expand data	2 bytes
SAONT package	4160 bytes	RAONT package	4130bytes	AONT package	4130 bytes
Each share	416 bytes	Each share	413bytes	Each share	413bytes
Total storage	6.5KB	Total storage	6.45KB	Total storage	6.45KB

K_A to 64 bytes. The 64-byte hash value is XORed with a 64-byte expanded key to produce a 64-byte cipher block S_{r+1}. The ciphertext S and a cipher block S_{r+1} are combined into 4160-byte SAONT package.

To disperse the SAONT package with the threshold (10, 16) RS erasure code, the 4160 bytes SAONT package is split into 10 shares stored in 10 nodes. There are 6 additional shares calculated using the RS code algorithm. Hence, the total storage is $416*16 = 6.5KB$. The storage of AONT-RS is 6.45KB based on a 4 KB data block and the threshold (10, 16) RS erasure code, described in [21]. The storage space of RAONT-RS is 6.45KB and is the same as AONT-RS because RAONT-RS removes the canary and uses a 32-byte random key and hash value. Table 2 presents the comparison between AONT-RS and RAONT-RS; the storage space of SAONT-RS is slightly bigger than AONT-RS and RAONT-RS. The reason is that the 512-bit hash value is used in SAONT-RS, which makes the length of cipher block S_{r+1} longer than value b in AONT-RS and RAONT-RS.

C. DICUSSION

We carry out the experiments for two scenarios on efficiency and test storage on an SAONT-RS system. By contrast, with the efficiency of SAONT-RS with AONT-RS and RAONT-RS, the test results show that in the case of an existing DoS attack, such as an attack that continually damages the node data in the reconstruction phase, the efficiency of the SAONT-RS system is better than the AONT-RS scheme because it can prevent the DoS attack. By comparing the storage of SAONT-RS with AONT-RS and RAONT-RS, the test results show that the SAONT-RS scheme requires slightly more storage than AONT-RS, which is not an issue, in general. SAONT-RS is an efficient distributed storage scheme.

VII. CONCLUSION

Distributed storage provides storage services to the outside world by computing data blocks in different nodes throughout a network. With the rapid development and application of distributed storage, people have focused on its security. AONT-RS and RAONT-RS are the typical algorithms in distributed storage. In the study, we found it has four security vulnerabilities. First, correct recovery is not efficient, so it could be limited in adoption due to its computational demands. Second, it is largely unable to prevent short plaintext attacks. Moreover, it fails to prevent DoS attacks, such as an attack that continually damages the node data in the reconstruction phase. Finally, it cannot correctly recover the node data damaged by the attacker. Hence, to address the vulnerabilities, we propose a secure and efficient distributed storage scheme SAONT-RS based on an SAONT proposed by us and an RS erasure code. SAONT-RS calculates a 512-bit hash value using the SHA-3 function and expands the random key to the length of the hash value to largely prevent the short plaintext attack. It uses SHA-3 to verify the integrity of the node data at the beginning of the reconstruction phase and combines RS erasure code to correctly and efficiently recover the damaged data and to prevent the DoS attack.

We analyze the secure efficient distributed storage scheme from confidentiality, anti-short plaintext attack, recoverability and integrity aspects, and evaluate the efficiency and storage. The results show that SAONT-RS achieves node data confidentiality, recoverability, integrity, anti-short plaintext attack, and good efficiency in distributed storage. SAONT-RS has great significance for cloud storage and big data with high security requirements. In the near future, we will improve system performance aiming at erasure coding under the condition of satisfying high security on distributed storage.

REFERENCES

[1] Z.-S. Feng, Z.-Z. Qin, and D. Yuan, "Techniques of secure storage for cloud data," *Chin. J. Comput.*, vol. 38, no. 1, pp. 150–163, Jan. 2015.

[2] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[3] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li, "Cooperative recovery of distributed storage systems from multiple losses with network coding," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 268–276, Feb. 2010.

[4] F. Chang *et al.*, "Bigtable: A distributed storage system for structured data," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, Jun. 2008, Art. no. 4.

[5] Y.-J. Wang, W.-D. Sun, S. Zhou, X.-Q. Pei, and X.-Y. Li, "Key technologies of distributed storage for cloud computing," *J. Softw.*, vol. 23, no. 4, pp. 962–986, Apr. 2012.

[6] Y. Zhang, R. Power, S.-Y. Zhou, Y. Sovran, M. K. Aguilera, and J. Y. Li, "Transaction chains: Achieving serializability with low latency in geo-distributed storage systems," in *Proc. 24th ACM Symp. Oper. Syst. Princ.*, Farminton, PA, USA, Nov. 2013, pp. 276–291.

[7] V. L. Kher and Y. Kim, "Securing distributed storage: Challenges, techniques, and systems," in *Proc. ACM workshop Storage Secur. Survivability*, Fairfax, VA, USA, Nov. 2005, pp. 9–25.

[8] H.-G. Zhang, W.-B. Han, X.-J. Lai, D.-D. Lin, J.-F. Ma, and J.-H. Li, "Summary of cyberspace security," *Sci. Sinica Inf.*, vol. 46, no. 2, pp. 125–164, Jan. 2016.

[9] Thestreet. (Jun. 14, 2018). *Millions of Records From Dun & Bradstreet Database Leaked*. [Online]. Available: https://www.thestreet.com/story/14043900/1/

[10] Medianama. (Jun. 14, 2018). *McDonald's India App May Have Leaked Personal Info of 2.2M Users*. [Online]. Available: https://www.medianama.com/2017/03/223-mcdonalds-india-data-leak/

[11] H. Li, W.-H. Sun, F.-H. Li, and B.-Y. Wang, "Secure and privacy-preserving data storage service in public cloud," *J. Comput. Res. Develop.*, vol. 51, no. 7, pp. 1397–1409, Jan. 2014.

[12] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 60, no. 1, pp. 212–236, Jan. 2014.

[13] K. K. Mar, Z. Hu, C. Y. Law, and M. Wang, "Securing cloud data using information dispersal," in *Proc. 14th Annu. Conf. Privacy, Secur. Trust (PST)*, Auckland, New Zealand, Dec. 2016, pp. 445–448.

[14] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[15] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM*, vol. 36, no. 2, pp. 335–348, Apr. 1989.

[16] Z.-H. Tan, G.-M. Yang, X.-W. Wang, W. Cheng, and J.-Y. Ning, "Threshold secret sharing scheme based on multidimensional sphere for cloud storage," *J. Softw.*, vol. 27, no. 11, pp. 2912–2928, Nov. 2016.

[17] J. Xu, J. Li, J. Han, F.-X. Li, and F.-C. Zhou, "Secure cloud storage model based on threshold public key encryption and erasure codes over exponents," *J. Softw.*, vol. 27, no. 6, pp. 1463–1474, Jun. 2016.

[18] H. Lahkar and C. R. Manjunath, "Towards high security and fault tolerant dispersed storage system with optimized information dispersal algorithm," *Int. J. Adv. Res. Comput. Sci. Technol.*, vol. 2, no. 3, pp. 71–75, Jul. 2014.

[19] US-CERT. (Jan. 20, 2018). *Understanding Denial-of-Service Attacks*. [Online]. Available: https://www.us-cert.gov/ncas/tips/ST04-015

[20] L. Chen, T. M. Laing, and M. Keith, "Revisiting and extending the AONT-RS scheme: A robust computationally secure secret sharing scheme," in *Proc. 9th Int. Conf. Cryptol. Inf. Secur.*, Dakar, Senegal, Apr. 2017, pp. 40–57.

[21] J. K. Resch and J. S. Plank, "AONT-RS: Blending security and performance in dispersed storage systems," in *Proc. FAST 9th Usenix Conf. File Storage Technol.*, Feb. 2011, pp. 191–202.

[22] P. Rogaway and M. Bellare, "Robust computational secret sharing and a unified account of classical secret-sharing goals," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, 2007, pp. 172–184.

[23] Y.-J. Wang, F.-L. Xu, and X.-Q. Pei, "Research on erasure code-based fault-tolerant technology for distributed storage," *Chin. J. Comput.*, vol. 40, no. 1, pp. 236–255, Jan. 2017.

[24] X.-H. Luo and J.-W. Shu, "Summary of research for erasure code in storage system," *J. Comput. Res. Develop.*, vol. 49, no. 1, pp. 1–11, 2012.

[25] H.-G. Rong, J.-X. Mo, B.-G. Chang, G. Sun, and F. Long, "Key distribution and recovery algorithm based on Shamir's secret sharing," *J. Commun.*, vol. 36, no. 3, pp. 60–69, Mar. 2015.

[26] Z.-X. Fu, G. Shen, B. Li, and B. Yu, "Threshold multi-secret visual cryptography scheme with perfect recovery," *J. Softw.*, vol. 26, no. 7, pp. 1757–1771, Jul. 2015.

[27] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 220–232, Apr./Jun. 2012.

[28] M. Baldi, N. Maturo, E. Montali, and F. Chiaraluce, "AONT-LT: A data protection scheme for cloud and cooperative storage systems," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Bologna, Italy, May 2014, pp. 566–571.

[29] M. Li, C. Qin, P. P. C. Lee, and J. Li, "Convergent dispersal: Toward storage-efficient security in a cloud-of-clouds," in *Proc. 6th USENIX Conf. Hot Topics Storage File Syst.*, Philadelphia, PA, USA, Jun. 2014, pp. 1–5.

[30] L. Yang and X. Lu, "An efficient dispersal storage scheme based on Ring-LWE and NTT," in *Proc. 12th Asia Joint Conf. Inf. Secur. (AsiaJCIS)*, Seoul, South Korea, Aug. 2017, pp. 23–30.

[31] M. Li, C. Qin, J. Li, and P. P. C. Lee, "CDStore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal," *IEEE Internet Comput.*, vol. 20, no. 3, pp. 45–53, May/Jun. 2016.

[32] G. Bian, S. Gao, and B. Zhao, "Security structure of cloud storage based on dispersal," *J. Xi'an Jiaotong Univ.*, vol. 45, no. 4, pp. 41–45, Apr. 2011.

[33] S. J. Lin and W. H. Chung, "An efficient (n, k) information dispersal algorithm for high code rate system over fermat fields," *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2036–2039, Dec. 2012.

[34] H. Krawczyk, "Secret sharing made short," in *Proc. Annu. Int. Cryptol. Conf.* Springer, Aug. 1993, pp. 136–146.

[35] R. L. Rivest, "All-or-nothing encryption and the package transform," in *Proc. 4th Int. Workshop Fast Softw. Encryption*, Jan. 1997, pp. 210–218.

[36] A. Abreha and S. Shetty. (Apr. 10, 2017). *Erasure Coding and AONT Algorithm Selection for Secure Distributed Storage*. Accessed: Apr. 2017. [Online]. Available: http://www.ece.gmu.edu/

[37] A. Rickus, E. Pfluegel, and N. Atkins, "Chaos-based image encryption using an AONT mode of operation," in *Proc. Int. Conf. Cyber Situational Awareness, Data Anal. Assessment (CyberSA)*, London, U.K., Jun. 2015, pp. 1–5.

[38] J. Shen, J. Gu, Y. Zhou, and X. Wang, "Cloud-of-clouds storage made efficient: A pipeline-based approach," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2016, pp. 724–727.

[39] A. M. Ahmadian and M. Amirmazlaghani, "Computationally secure secret image sharing," in *Proc. Iranian Conf. Electr. Eng. (ICEE)*, Tehran, Iran, May 2017, pp. 2217–2222.

[40] R. Seiger, S. Groß, and A. Schill, "SecCSIE: A secure cloud storage integrator for enterprises," in *Proc. IEEE 13th Conf. Commerce Enterprise Comput.*, Luxembourg, Sep. 2011, pp. 252–255.

[41] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[42] Google. (Jun. 3, 2018). *The First Concrete Collision Attack Against SHA-1*. [Online]. Available: https://shattered.io/

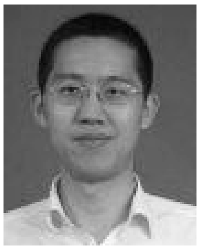
[43] W. Stallings, "Cryptographic Hash Functions," in *Cryptography and Network Security: Principles and Practice*, 7th ed. Upper Saddle River, NJ, USA: Pearson, 2017, pp. 313–351.

[44] National Institute of Standards and Technology. (Sep. 27, 2017). *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

LILI YAO was born in China in 1993. She is currently pursuing the master's degree with the School of Computer Science, South Central University for Nationalities, China. Her current research interests include data storage security and protocol security.

JINTIAN LU was born in China in 1991. He is currently pursuing the master's degree with the School of Computer, South Central University for Nationalities, China. His current research interests include the formal and inverse analysis of security protocols.

JIABING LIU was born in 1993. He is currently pursuing the master's degree with the School of Computer, South Central University for Nationalities, China. His current research interest is the formal analysis of security protocol.



DEJUN WANG was born in 1974. He received the Ph.D. degree in information security from Wuhan University, China. He is currently an Associate Professor with the School of Computer, South Central University for Nationalities, China. He has authored or co-authored over 20 papers in international/national journals and conferences. His current research interests include security protocols and formal methods.

BO MENG was born in China in 1974. He received the M.S. degree in computer science and technology in 2000 and the Ph.D. degree in traffic information engineering and control from the Wuhan University of Technology, Wuhan, China, in 2003. From 2004 to 2006, he was with Wuhan University as a Post-Doctoral Researcher in information security. From 2014 to 2015, he was with the University of South Carolina as a Visiting Scholar. He is currently a Full Professor with the School of Computer Science, South Central University for Nationalities, China. He has authored or co-authored over 50 papers in International/National journals and conferences. In addition, he has also published two books *Automatic Generation and Verification of Security Protocol Implementations* and *Secure Remote Voting Protocol* (China: Science Press). His current research interests include big data security, blockchain, security protocols, and formal methods.