RAON-RB: A Verifiable Randomized Non-Separable Encryption Scheme for Secure Cloud Storage

Fatty M. Salem
Department of Electronics and Communications,
Faculty of Engineering, Helwan University
1 Sherif St., Helwan, Cairo, Egypt

Mohamed Osman
Department of Computers,
Faculty of Engineering, Helwan University
1 Sherif St., Helwan, Cairo, Egypt

ABSTRACT
Exponential increase in data inside endeavors is making a gush in the storage requirements and its security. Typical encrypting techniques of separable nature suffer against powerful adversaries due to the exponential improvement of computing power. Therefore, some cloud storage systems apply erasure coding in addition to encryption to prevent adversaries from revealing or/and controlling stored information. Nevertheless, still a powerful adversary can extract useful information from a compromised server. In this paper, a new storage scheme is introduced implying an All-Or-Nothing (AON) encryption mode to separately randomize the encrypted blocks and hashing the resulted pseudorandom blocks altogether afterwards, and with the aid of the salting technique, called Replicated BYTE, integrity is ensured. The scheme will be called Randomized AON plus Replicated BYTE (RAON-RB). By employing systematic Reed Solomon erasure coding, the proposed scheme will be more applicable to distributed systems. The proposed scheme is secure even if all but one storage servers have been compromised and even if the encryption key is disclosed. Moreover, to resist share modification and localize faulty server(s), the proposed scheme verifies the received shares before they are involved in the reconstruction process, and hence, the scheme can save superfluous computations as well.

Keywords
Cloud computing, distributed storage, AON encryption, non-separable block cipher, Reed Solomon erasure code, hashing.

1. INTRODUCTION
Cloud computing is considered one of today's most tempting technology areas that provides for convenient on-demand utilization of a huge pool of resources ranging from physical infrastructure to higher-level applications. The temptation comes from its cost efficiency and flexibility [1]. Our field of concern will be on long-term archival storage service offered by Cloud Service Providers (CSP) to store and retrieve enterprise data. Clearly, the sensitivity of stored information urged the CSPs to compete for providing storage pools with increased levels of security and reliability. The traditional technique for securing cloud data is via encrypted replicas of user's data which is still used by recent systems [2, 3]. This technique is not space-efficient as the required storage space is multiplied by the number of copies. Moreover, each redundant mirror represents an additional attack station for adversary. Therefore, storing an entire copy of data at several storage locations compels that the administrator has to provide each storing node with the same expensive security measures.

Studies show that fragmentation increases reliability rather than replication [4-7]. To illustrate this assertion, the reliability formula is refered in [8] that calculates the reliability at a given instant of time in a distributed environment;

$$P = \sum_{i=0}^{rf} \frac{\binom{m}{i}\binom{n-m}{f-i}}{\binom{n}{f}}$$

where P is the probability that a document is available, n the number of machines, m the number of currently unavailable machines, f the number of fragments per document, and rf is the maximum number of unavailable fragments that still allows the document to be retrieved. For instance, with a million machines, ten percent of which are currently down, simple replication without erasure codes provides only two nines (0.99) of reliability. A 1/2-rate erasure coding of a document into 16 fragments gives the document over five nines of reliability (0.999994), yet consumes the same amount of storage. With 32 fragments, the reliability increases by another factor of 4000 [8]. Fragmentation also improves security by prohibiting an attacker from getting hold of the entire entity of the encrypted data by leveraging dispersal using erasure coding techniques such as Information Dispersal Algorithm (IDA) and Reed Solomon codes. Dispersal could be done across servers of the same cloud or through a-cloud-of-clouds model (also called an intercloud) [9-12]. Many cloud storage systems employ dispersal algorithms like POTSHARDS [13], OceanStore [8], and Clevesafe [14]. Despite former advantages, straightforward dispersal techniques encounter security flaws; therefore some techniques apply encryption to data beforehand. Nonetheless, a powerful attacker that is able to employ large processing capabilities can extract information from encrypted data shares on a single compromised server. Therefore the need has emerged for inseparable encryption modes which reveal no information from less than all encrypted pieces.

In this paper, a new storage scheme is introduced that uses an All-Or-Nothing (AON) encryption mode by separately randomizing each encrypted block and then hashes the resulted pseudorandom blocks altogether. The resulted pseudorandom message is then encoded and distributed via systematic Reed Solomon coding [15]. The scheme is also robust against share modification via the proposed space-efficient integrity verification technique that can detect and exclude corrupted shares before key reconstruction, and hence, can safe costly computations.

The rest of the paper is organized as follows. In Section 2, the previous proposed work on secure cloud storage is discussed. In Section 3, a brief description of semantic security is provided. In Section 4, the used system model is depicted. The proposed scheme is presented in Section 5. The proposed scheme is evaluated in Section 6. Section 7 is devoted to the performance evaluation. Finally, the paper is concluded in Section 8.

2. PREVIOUS WORK

Many schemes have been proposed for secure data storage in the cloud and a number of these schemes are based on secret splitting. POTSHARDS [13] performs two levels of secret splitting: the first level is an XOR-based splitting algorithm that is tuned for secrecy by producing *n* fragments from a data object, the second split level uses Shamir secret sharing (SSS) [16] to construct a list of shards from a fragment and distribute them securely among the archives which provides perfect secrecy. Encryption is avoided and hence no key management required. Both XOR-based split and Shamir secret sharing are computationally expensive for encoding and decoding large files in addition to high storage cost; a two-way XOR split followed by a (2,3) secret split increases storage requirements by a factor of six [13]. POTSHARDS utilizes 4-byte algebraic signatures [17] for remote integrity checking with no need for downloading the stored shards. Algebraic signatures have the property that the signature of the parity equals the parity of signatures of corresponding shards. It was stated in [17] that algebraic signatures are similar to "cryptographically secure" hash functions such as MD5, SHA-1, and SHA-256, though algebraic signatures are not cryptographically secure because it is easy to deliberately construct two strings that have the same algebraic signature.

OceanStore [8] provides deep archival storage. For archival, versions of stored objects are encoded by employing erasure codes such as Reed-Solomon codes and are spread over hundreds or thousands of servers. This addresses the problem of the increased overhead resulted in Shamir's Secret Sharing. However, erasure coding does not provide any security guarantees when subjected to a powerful attacker compromising a number of shares less than the reconstruction threshold. DepSKY-CA [18] utilizes Secret Sharing Made Short (SSMS) of Krawczyk [19] to provide improved security. In DepSKY-CA, data is encrypted using AES encryption and then encoded using Information Dispersal Algorithm (IDA) into *n* shares. The encryption key is encoded and distributed using Shamir Secret Sharing. The encrypted file slices are distributed over *n* storage servers. For integrity in DepSKY-CA, the user generates and stores *n* digests for the *n* dispersed shares in the metadata file at user side.

To make launching a brute force attack harder, Rivest [20] suggested an All-Or-Nothing encryption mode that implies three rounds of encryption; a package transform encryption round, a fixed-public-key (k_0) encryption round for embedding the package transform key into the pseudo-random output blocks, and finally a round of outer encryption block cipher using a different key which is kept secret. This constitutes a non-separable CPA-secure AON encryption mode that cannot be inverted unless all ciphertext blocks are acquired. Rivest AON encryption incurs a large computation overhead which is unacceptable when storing large files in the cloud environment (the legitimate communicants pay a penalty of approximately a factor of three in the time it takes them to encrypt or decrypt in all-or-nothing encryption mode compared to an ordinary separable encryption mode [20]). Also the need for the management and storage of an outer encryption secret key is considered an added burden. A simpler variant to Rivest AONT approach was proposed by Resch et al. to enrich Rabin's IDA with confidentiality called AONT-RS [21]. AONT-RS is adopted by Cleversafe [14], a recent cloud service provider, in their dispersed storage system. No outer encryption is involved in AONT-RS. The output pseudo-random message is dispersed using a systematic Reed Solomon erasure code for the sake of fault tolerance and added security. AONT-RS is still susceptible to brute-force attacks by an attacker that has compromised one of the first *m* storage servers (pure ciphertext). PAST [2], a global-scale storage system of read-only data, preserves integrity by replication which is not space optimal. Integrity in Rivest AONT [20] and AONT-RS [21] is verified using a predetermined block, called CANARY value in [21], appended to the plaintext before pre-processing. This provides detection of distorted plaintext after the inverse-AONT has been performed having the user priorly execute all the expensive decryption and inversion calculations before corruption is detected, which is considered inefficient in terms of computation costs. This also doesn't localize the faulty/malicious servers. Also the fixed known canary value will make an ideal means for a known-plaintext attacker to check the encryption key against the corresponding ciphertext block.

Another integrity assurance technique which is called distributed fingerprints was suggested by Krawczyk [22]. This technique calculates public fingerprints for each share and then shares each one, using error correcting codes, over the *n* storage servers. These codes can reconstruct shared fingerprints from *t* altered pieces but at the expense of a large blowup factor of at least $(n/(n-2t))$. This technique is much simpler than general signature schemes but also incurs a large space overhead of $(n^2/(n-2t)*|\text{fingerprint}|$, in addition to complex computation overhead at decoding time. The authors in [23] proposed a mechanism for partitioning data in a multi-cloud system at a lower time overhead than classical cryptographic techniques. The processing of this scheme on data shredding combined with a data pattern elimination technique. Later, the authors in [24] have introduced a keyless efficient algorithm for data protection by means of fragmentation that overcome the data pattern problem. A modified version of AONT-RS called Convergent AONT-RS technique (CAONT-RS) is introduced in [25, 26] which allows fragments' deduplication. CAONT-RS divides initial data into pieces of variable size. Pieces are then transformed into *n* fragments using the CAONT-RS technique, *k* pieces are required for data recovery. CAONT-RS is similar to the AONT-RS technique, except that the processing step uses optimal asymmetric encryption padding (OAEP)[27] instead of a block cipher encryption. Moreover, in CAONT-RS, a cryptographic hash generated from the initial data is used instead of the random key of the AONT transform. Furthermore, the authors in [28] have introduced a technique of data ownership challenge and proxy re-encryption (PRE) to manage encrypted data storage with de-duplication.

Additionally, many auditing techniques are proposed to effectively check the integrity of the stored data. The user (data owner) can check the integrity of the stored data based on two-party auditing [29] where the user and the service provider are involved. However, the proposed schemes may not be appropriate of auditing as both the user and the service provider are not able to provide unbiased result. Hence, a Third-Party auditing based schemes are proposed where the thirds party auditor takes the responsibility of convincing both the user and the service provider [30, 31]. Recently, A

dynamic outsourced data auditing scheme was proposed [32] to provide verifiable dynamic updates to outsourced data based on batch leaves authenticated Merkle Hash Tree (BLA-MHT) for batch verify multiple leaf nodes and their own indexes all together. Furthermore, an identity-based data outsourcing scheme is proposed in [33] where the authorized proxy only allows the next process and outsources the file. However, the file integrity is verified by the public auditor.

In summary, the secure storage in cloud without replication using fragmentation, the file integrity, and detecting the faulty server before executing the costly computations are the goals of this paper.

3. SEMANTIC SECURITY

Semantic Security [34] means that under an unknown key, an adversary cannot derive any information about the plaintext (other than the length of the message) over what the adversary would have known without seeing the ciphertext, or some ciphertext. The notion is also referred to as indistinguishability of encryptions and noted as IND.

If a Probabilistic Polynomial-time (PPT) adversary is allowed to query an encryption oracle Enc(), which runs in a reasonable amount of times, and submit two equal-sized plaintext messages (m_0, m_1) to a challenger for encryption and receives c. The cryptosystem is said to be indistinguishable under adaptive chosen plaintext attack if the adversary has only a negligible advantage over random guessing in distinguishing the plaintext message m_i that corresponds to the received ciphertext c. Figure 1 shows the IND-CPA game described as follows:

1. The adversary A may perform any number of encryptions or other operations with the encryption oracle Enc().
2. The adversary submits two distinct chosen plaintexts m_0, m_1 to the challenger C.
3. The challenger selects a bit b:(0, 1) uniformly at random, and sends the ciphertext c of m_b back to the adversary.
4. The adversary is free to perform any number of additional computations or encryptions.
5. Finally, the adversary outputs a guess b' for the value of b.

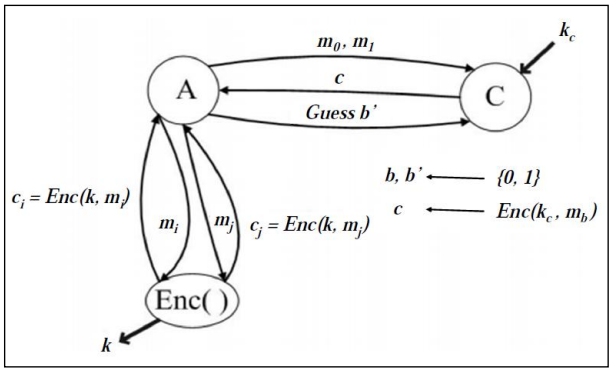

Fig 1: IND-CPA game

4. THE MODEL

The proposed distributed cloud system is constituted of two entities: the user, denoted U, and n storage servers, denoted $\{V_1,\ldots, V_n\}$. It is structured as a client user communicating with a number of independent archives, and it is assumed that each server appropriately authenticates the user. the distributed model is inspired by the storage model that was set in [19]. In this case, in addition to the threshold parameter m of shares needed to reconstruct the secret, a bound t on the number of malicious servers is to be specified. It is necessary that $t < m$ (a coalition of only malicious parties cannot reconstruct the secret), and $m \leq n- t$ (there are enough good parties to reconstruct the secret). These two relations imply $2t+1 \leq n$, i.e. a majority of honest parties is required. The communication between the user and the storage servers is supposed to be through a web environment of completely connected secure and authenticated links which can be realized using SSL.

5. THE PROPOSED SCHEME

In this section, the proposed Randomized AON plus Replicated BYTE (RAON-RB) scheme for cloud storage systems is presented which provides CPA-security against modern powerful attackers that are able to employ large processing capabilities in order for them to launch computationally-heavy attacks. The proposed scheme is consisted of two parts: Randomized AON (RAON) and Replicated BYTE (RB).

5.1 Randomized AON

The randomized AON uses a suitable underlying block cipher $\mathcal{E}$ in the counter (CTR) mode for encrypting user's file f before the ciphertext C is processed using a linear post-transform that outputs the pseudo-ciphertxt C'. The pseudo-ciphertxt, padded to the encryption key k, is then distributed over n storage servers using systematic Reed Solomon erasure coding [15] for fault-tolerance and increased security.

As there's no key-exchange burden between legitimate users, each file f can have a unique encryption key k, which is close to perfect secrecy of One-Time Padding (OTP) [35]. As shown in Figure 2, the randomized AON works as follows:

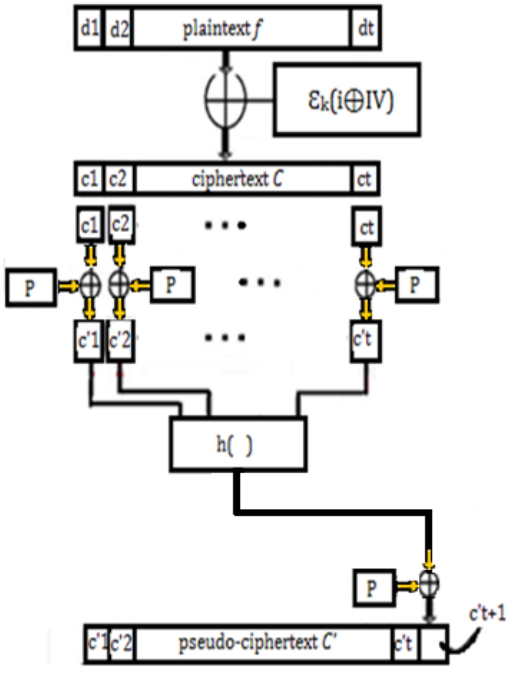

Fig 2: The Randomized AON.

- On input a plaintext bit-stream f of size b bytes, its is divided into t blocks d_i; $i = 1, 2\ldots, t$.
- A random *nonce* value IV picked from the space $\{0, 1\}^{|di|}$ is XORed to each counter value i to form the random CTR encryption mode block.

- The plaintext *f* is encrypted using a randomly chosen key *k* into ciphertext *C* of *t* blocks c_i:

  $c_i = d_i \oplus \varepsilon_k(i \oplus IV)$

- Calculate *h*(*C*).
- Calculate the randomizing block P:

  $P = IV \oplus h(C)$

- Randomize the ciphertext blocks c_i to:

  $c_i' = c_i \oplus P$

- Calculate $h(c_1' \| c_2' \| ... \| c_t')$.

- Impede P in an additional block c_{t+1}':

  $c_{t+1}' = P \oplus h(c_1' \| c_2' \| ... \| c_t')$

- The Pseudo-ciphertext C' will be the sequence:

  $C' = h(c_1' \| c_2' \| ... \| c_t' \| c_{t+1}')$

k is appended to *C'* as c'_{t+2}, and then the output blocks are encoded into *n* shares S_i via systematic erasure coding [15] across *n* storage servers. Figure 3 illustrates the systematic erasure coding, where the first *m* slices are generated by splitting the Pseudo- ciphertext package, the rest are generated using the generator matrix [21].

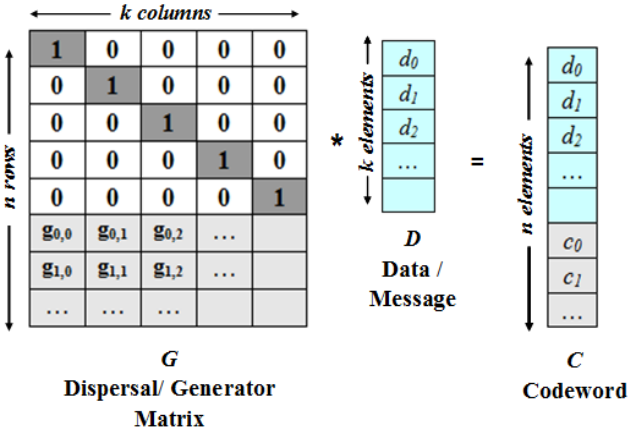

Fig 3: The systematic erasure coding

Inverting the RAON:
After collecting the distributed shares, the user U reconstructs *C'*‖*k*, extracts *k*, and then:

- Calculates $h(c_1' \| c_2' \| ... \| c_t')$.
- Calculates $P = h(c_1' \| c_2' \| ... \| c_t') \oplus c_{t+1}'$.
- Calculates ciphertext blocks $c_i = c_i' \oplus P$.
- Calculates *h*(*C*).
- Calculates $IV = h(C) \oplus P$.
- Given *k*, decrypts c_i into d_i:

  $d_i = c_i \oplus \varepsilon_k(i \oplus IV)$.

5.2 Replicated BYTE (RB)

While performing the retrieval process of the stored shares S_i, it's possible that some malicious servers return modified shares in order to prevent the reconstruction process. The other related solutions for integrity whether introduce increased overhead in terms of storage space, communication, and computation, or use costly cryptographic tools that degrade the efficiency in the cloud environment. In this section, a new simple integrity assurance algorithm is proposed with the help of the *salting* technique, called Replicated BYTE (RB), which ensures the integrity of the stored shares as long as a majority of honest storage servers exists. The algorithm is shown in Figure 4. It doesn't use costly cryptographic tools, and is also space efficient as it adds up a small overhead to the stored share. The simple MD5 hash function (message digest) is leveraged in order to evaluate the correctness of the received share S'_i. The *salting* technique is also employed, used in securing passwords, for the purpose of blinding.

Before storing share S_i at server V_i, user U performs the following steps:

- Picks a *salt* value of random length ≤ $|d_i|$ randomly.
- Appends the *salt* value to S_i;

  $T_i = S_i \| salt$

- Calculates *h*(T_i).
- Splits *h*(T_i) into *j* bytes b_{ij}.
- For each share S_i, calculates $BYTE_i$:

  $BYTE_i = b_{i1} \oplus b_{i2} \oplus ... \oplus b_{ij}$

- Concatenates the bytes in a block:

  $BLOCK = (BYTE_1 \| BYTE_2 \| ..... \| BYTE_n)$

Replicates the *n*-byte *BLOCK* for *n* times and then appends each replica to a share S_i, and sends (*BLOCK* ‖ S_i) to the corresponding storage server V_i.

- *Salt* block is secretly stored at user side.

Verification:
Upon sending a request for retrieval of file *f* to the storage servers by its owner, each server V_i will return a share S'_i along with an n-byte *BLOCK'*. User *U* proceeds as follows:

- Performs a *majority* voting between the *n* received *BLOCK'* values which leads to a legitimate *BLOCK* value.
- Dismantles *BLOCK* into its *n* constituting $BYTE_i$ values.
- Appends the stored *salt* value to S'_i.
- Calculates $BYTE'_i$ for received share S'_i as in Figure 4.
- If $BYTE'_i \neq BYTE_i$, then the user discards the corresponding share S'_i and declares that server V_i is malicious. Otherwise, the share is correct and can be used in reconstruction.

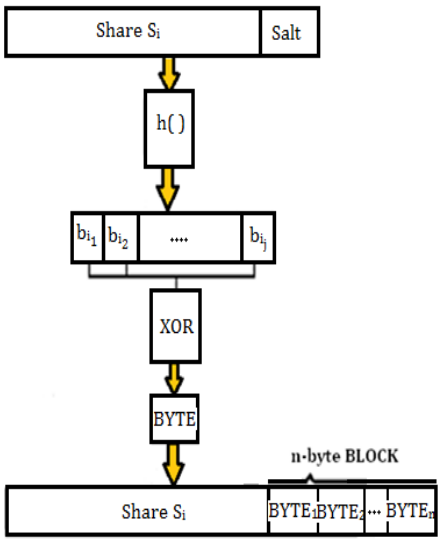

Fig 4: Replicated BYTE (RB).

6. EVALUATION

In this section, the storage security and integrity provided by the proposed scheme are evaluated.

6.1 Storage Security

Based on the semantic security of the underlying CTR encryption mode, using a large enough counter block size (128 bits) allows the (counter, nonce) value to take 2^{128} unique permutations. This nature of the proposed CTR encryption provides perfect security similar to One Time Padding (OTP) while avoiding its great mathematical computations. When performing the game explained in section 4, an attacker will get no more than a negligible advantage over, randomly, guessing the related counter/nonce combination. Even if the encryption key *k* is exposed, the security of the scheme is maintained because the *IV* value that randomizes the counter pattern in the CTR encryption mode is still anonymous to the attacker. Therefore, the scheme is IND-CPA.

Now, the proposed scheme will be proved t be non-separable, as an attacker with less than t+1 output blocks, c'_i, will not get information about even one ciphertext block c_i.

Proof: Note that while inverting the transform, each ciphertext block c_i is fully dependent on all t+1 pseudo-ciphertext blocks of C'; $c_i = c_i' \oplus (c_{t+1}' \oplus h(c_1' \| c_2' \| ... \| c_t'))$, for $1 \le i \le t$. Therefore, given any t blocks of C', then c_i could take any of $2^{|c_i'|}$ possible values (random guessing) which proclaims the inseparability of the proposed scheme.

Combining RAON with a systematic erasure coding algorithm, along with the wise choice of *m* and *n*, creates a system suitable for cloud implementation and provides fault-tolerance capability with less cost than replication, in addition to increasing security by preventing an adversary from acquiring all pseudo-ciphertext blocks. Dispersing the encryption key along with the pseudo-random output does not affect the security of the proposed scheme, as it is still secure even with the key *k* exposed. Dispersion step is performed for the sake of availability.

6.2 Integrity

The security of the Replicated BYTE (RB) algorithm and how it can maintain the file *f* intact with the help of the secure salted hash technique will be proved. Suppose a corrupted storage server V_i wants to undetectably fool the data owner and return a mutilated slice S'_i, it must produce the same $BYTE_i$ value. As length and content of *salt* value is anonymous to V_i, an adversary has only a negligible advantage over random guessing through all possible *salt* values in order to find a false combination $S'_i \| salt$ that produces the same $BYTE_i$. Additionally, the attacker is not able to undergo cryptanalytic calculations on the MD5 fingerprint in order to compromise its pre-image resistance (given a hash *h*, it should be difficult to find any message X such that h = hash(X)) and deduct T_i, as it is obtainable to the attacker because it is not stored anywhere; only the bitwise difference $BYTE_i$ of $h(T_i)$ is stored at server V_i.

7. PERFORMANCE COMPARISON

The performance comparison of the proposed storage scheme against other related all-or-nothing schemes will be in terms of speed, storage space, security, and integrity. The propose RAON-RB scheme will be compared with both Rivest AON and AONT-RS techniques. The comparison will involve the performance of the suggested transform before dispersal, as erasure code is added only for fault tolerance purposes. POTSHARDS [13] uses secret splitting which is expensive for storing large files in terms of computation and storage costs. Also, both OceanStore [8] and DepSKY [18] suffer from erasure code security weakness.

The open source C++ codes are used for cryptographic primitives that were implied in the compared schemes. These codes are compiled with Microsoft Visual C++ 2005 SP1, and ran them on an Intel Core 2 1.83 GHz processor under Windows 7 in 32-bit mode, for speed evaluation. The measured encoding rates for encryption modes and hashing functions that were employed in the compared schemes are shown in Table 1. RAON produces a pseudo-ciphertext message by applying *one* round of AES-128/CTR encryption plus two MD5 hashing functions over $(t * | d_i |)$ size of data.

Table 1. Encoding rates.

Algorithm	Encoding rate (MiB/s)
AES-128/CTR	139
AES-128/ECB	109
MD5	255

Using the data obtained in Table 1, the RAON total encoding time = $\frac{t * | d_i | (\text{in MiB})}{66}.5$.

The total encoding rates of Rivest AON encryption, RAON, and AONT-RS scheme will be compared. The total encoding rate achieved by RAON is found to be 66.5 MiB/s. Rivest AON mode executes three rounds of AES-128/ECB (all or nothing codebook mode) which totalizes to 36.3 MiB/s encoding rate. AONT-RS has a total encoding rate of 54.5 MiB/s. Table 2. shows the comparison between existing AON storage schemes in various aspects.

The proposed scheme is fully parallelizable due to applying CTR encryption; computing blocks (c_1, c_2,..., c_t) can be implemented at the same time. Concluded from encoding rate calculations and former security analysis, it can be stated that RAON technique outperforms both Rivest AON [20] and AONT-RS [21] in the field of confidentiality. In the field of integrity assurance, The Replicated BYTE (RB) algorithm adds storage overhead to the system as n^2 *bytes*. This is considered more than that of either Rivest AON or AONT-RS. However RB provides integrity verification prior to decoding the received shares, in contrast to [20] and [21] which both need the user to fully decode then decrypt before being able to determine the integrity state of the resulting plaintext. Moreover, RB localizes faulty servers, if exist, in order to avoid their corrupted shares before decoding. Distributed fingerprints scheme also achieves the same tasks as RB. However, RB outperforms distributed fingerprints of Krawczyk [22] in storage requirements and as it doesn't need the computationally expensive error correction technique.

To illustrate the advantage of the proposed RB scheme over distributed fingerprints of Krawczyk [22], suppose an example of a storage system of n=16 storage servers that needs to deal with the maximum number of (t=(n-1)/2) faulty parties. RB technique will cost 256 bytes of extra storage space while a distributed fingerprints scheme will require an overhead of 4 kbytes of storage space. Table 3. shows a comparison between RB and other integrity verification techniques. Algebraic signature technique is not cryptographically secure. Therefore, it was not included in this comparison.

Table 2. Comparison between RAON and other AON schemes

	Storage (Number of blocks)	Non-separable Security	Encoding Rate	Outer Key Safe Storage
Rivest AON	$(t+2)$ blocks	Non-separable	36.3 MiB/s	Required
AONT_RS	$n/m*(t+2)$ blocks	Separable	54.5 MiB/s	Not required
RAON	$n/m*(t+2)$ blocks	Non-separable	66.5 MiB/s	Not required

Table 3. Comparison between RB and other integrity verification techniques

	CSP-side Overhead	User-side Overhead	Faulty server localization	Verification step
Rivest AON	16 byte Canary	-	N/A	After fully decrypting user's file
AONT-RS	16 byte Canary	-	N/A	After fully decrypting user's file
RAON-RB	n^2 bytes	$\leq$ 16 bytes	Yes	Before decoding distributed shares
Distributed fingerprints	$n^2/(n-2t)$ x 16bytes	-	Yes	Before decoding distributed shares
DepSKY-CA	-	n^2 x 16 bytes	Yes	Before decoding distributed shares

8. ACKNOWLEDGMENTS

We would like to acknowledge with much appreciation our colleague Associate Professor Maged Hamada Ibrahim from Helwan University who provided insight and expertise that greatly assisted the research.

9. CONCLUSION

In this paper, the use of a variant All-Or-Nothing encryption mode plus systematic erasure coding is introduced to secure user data remotely stored in cloud environment. The proposed scheme offers several advantages: First, the resulting encryption mode is non-separable as the attacker will need to acquire nothing less than all the pseudorandom blocks in order for him to get useful information. Second, it consumes reasonable storage space and doesn't add expensive overhead on the storage servers; only two additional blocks with the information expansion ratio (IER) of the erasure technique. Third, the scheme needs no key management mechanism as the encryption key is simply padded then dispersed along with the pseudorandom blocks. Furthermore, the scheme remains secure even with the disclosure of the encryption key. Moreover, additional algorithm is introduced as part of the proposed storage scheme that checks on the shares stored outside of the user's control. It compresses a large share into individual byte. This compression saves storage space and network bandwidth. Additionally, blinding the stored shares using salting technique made it harder for a malicious server to forge its stored share. This allows inexpensive verification with relatively low storage costs. The combination of non-separable encryption, low storage cost, reasonable computation load, share dispersal, and resistance to malicious modification have made RAON-RB ideal for storing data on cloud environment.

10. REFERENCES

[1] Singh, R., Kumar, S. and Agrahari, S. 2012. Ensuring data storage security in cloud computing. IOSR Journal of Engineering, Vol. 2, Issue 12, pp. 17-21.

[2] Rowstron, A. and Druschel, P. 2001. Storage management and caching in past, a Large-scale, Persistent Peer-to-Peer Storage Utility. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01), Banff, Canada, pp. 188–201.

[3] Ghemawat, S., Gobioff, H. and Leung, S. T. 2011. The google file system. In 19th ACM Symposium on Operating Systems Principles, FAST, pp. 191–202.

[4] Tang, J. and Veijalainen, J. 1999. Using fragmentation to increase reliability for workflow systems. Journal of Integrated Design & Process Science, Vol. 3, No. 2, pp. 33-48.

[5] Zomaya, Y., Ahmad, I. and Khan, S. 2010. Comparison and analysis of ten static heuristics-based Internet data replication techniques. Journal of Parallel and Distributed Computing, Vol. 68, No. 2, pp. 113-136.

[6] Mei, A., Mancini, L. and Jajodia, S. 2013. Secure dynamic fragment and replica allocation in large-scale distributed file systems. IEEE Transactions on Parallel and Distributed Systems, Vol. 14, No. 9, pp. 885-896.

[7] Bilal, K., Khan, S. U., Zhang, L., Li, H., Hayat, K., Madani, S. A., Min-Allah, N., Wang, L. Chen, D., Iqbal, M., Xu, Z. and Zomaya, A. Y. 2015. DROPS: division and replication of data in cloud for optimal performance and security. Concurrency and Computation: Practice and Experience, Vol. 25, No. 12, pp. 1771-1783.

[8] Kubiatowicz, J., Bindel, D., Eaton, P., Chen, Y., Geels, D., Gummadi, R., Rhea, S., Weimer, W., Wells, C., Weatherspoon, H. and Zhao, B. 2000. Oceanstore: An architecture for globalscale persistent store. In Proceedings of ACM ASPLOS'2000, Cambridge, MA.

[9] Vukolic, M. 2010. The byzantine empire in the intercloud. ACM SIGACT News , Vol. 41, No. 3, pp. 105–111.

[10] Abu-Libdeh, H. Princehouse, L., and Weatherspoon H. 2010. RACS: A case for cloud storage diversity. In Proceedings of ACM SoCC.

[11] Cachin, C., Haas, R. and Vukoli´c, M. 2010. Dependable storage in the intercloud. IBM Research Report RZ 3783.

[12] Hu, Y., Chen, H., Lee, P. and Tang, Y. 2012. NCCloud: applying network coding for the storage repair in a cloud-of-clouds. In Proceedings of USENIX FAST.

[13] Storer, M. W., Greenan, K. M., Miller, E. L. and Voruganti, K. 2009. POTSHARDS—A secure, recoverable, long-term archival storage system. ACM Transactions on Storage Vol. 5, No. 2, Article 5.

[14] CLEVERSAFE, INC. 2014. Cleversafe dispersed storage. Community Portal: www.cleversafe.org.

[15] Moon, Y. S. 2011. Introduction to Reed-Solomon codes. Harvard University Department of Mathematics, August 2011.

[16] Shamir, A. 1979. How to share a secret. Communication of the ACM.

[17] Schwarz, S. J., T. and Miller, E. L. 2006. Store, forget, and check: using algebraic signatures to check remotely administered storage. In Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS ’06).

[18] Bessani, A. N., Correia, M. P., Quaresma, B. Andr´e, F. and Sousa, P. 2011. DepSky: dependable and secure storage in a cloud-of-clouds. In EuroSys, pp. 31–46.

[19] Krawczyk, H. 1994. Secret sharing made short. In CRYPTO ’93: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology, London, UK, pp. 136–146,

[20] Rivest, R. L. 1997. All-or-nothing encryption and the package transform. In Fast Software Encryption (FSE ’97), pp. 210–218.

[21] Resch, J. K. and Plank, J. S. 2011. AONT-RS: Blending security and performance in dispersed storage systems. In FAST, pp. 191–202.

[22] Krawczyk, H. 1993. Distributed fingerprints and secure information dispersal. In Prococeedings of the 13th ACM Symposium On Principles of Distributed Computation, ACM, New York. pp. 207–218.

[23] Cincilla, P., Boudguiga, A., Hadji, M. and Kaiser, A. 2015. Light blind: Why encrypt if you can share?. In 2015 12th International Joint Conference on e-Business and Telecommunications (ICETE), Vol. 04, pp. 361–368.

[24] Kapusta, K., Memmi, G. and Noura, H. 2016. Poster: A keyless efficient algorithm for data protection by means of fragmentation. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS ’16, New York, NY, USA, ACM, pp. 1745–1747.

[25] Boyko, V. 1999. On the security properties of OAEP as an all-or-nothing transform. In Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO ’99, London, UK, UK, Springer-Verlag, pp. 503–518.

[26] Li, M., Qin, C., Li, J. and Lee, P. P. C. 2016. CDstore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal. IEEE Internet Computing, Vol. 20, No. 3, pp. 45–53.

[27] Bellare, M. and Rogaway, P, 1995. Optimal asymmetric encryption. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 92–111.

[28] Yan, Z., Ding, W., Yu, X., Zhu, H. and Deng, R. H. 2016. Deduplication on encrypted big data in cloud. IEEE Transactions on Big Data, Vol.2, No.2, pp 138 - 150.

[29] T ajuddin, M. and Busi, K. C. 2013. An enhanced dynamic auditing protocol in cloud computing. International Journal of Engineering Trends and Technology, Vol. 4, Issue 7, pp. 3173- 3176.

[30] Wang, C., Wang, Q., Ren, K. and Lou, W. 2010. Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. INFOCOM, Proceedings IEEE, pp. 1-9.

[31] Shen, W., Yu, ., Xia, H., Zhang, H., Lu, X. and Hao, R. 2017. Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium. Journal of Network and Computer Applications, Vol. 82, pp. 56-64.

[32] Rao, L., Zhang, H. and Tu, T. 2017. Dynamic outsourced auditing services for cloud storage based on batch –leaves-authenticated merkle hash tree. IEEE Transactions on Services computing, Volume: PP, Issue: 99, pp. 1-14.

[33] Wang, Y., Wu, Q., Qin, B., Shi, W., Deng, R. H. and Hu, J. 2017. Identity-based data outsourcing with comprehensive auditing in clouds. IEEE Transactions on Information Forensics and Security, Vol. 12, No. 4, pp. 940-952.

[34] Watanabe, Y., Shikata, J. and Imai, H. 2003. Equivalence between semantic security and indistinguishability against chosen ciphertext attacks. In Proceedings of Public Key Cryptography-PKC 2003, Lecture Notes in Computer Science 2567, pp. 71-84.

[35] Menezes, A. J., Oorschot, P. and Vanstone, S. 1997. Handbook of Applied Cryptography. CRC Press Boca Raton, New York, London, Tokyo.