PAPER *Special Section on Security, Privacy and Anonymity of Internet of Things*

CloudS: A Multi-Cloud Storage System with Multi-Level Security

Lu SHEN[†], **Shifang FENG**[†], **Jinjin SUN**[†], **Zhongwei LI**[†], ***Nonmembers***, **Ming SU**[†], ***Member***, **Gang WANG**[†a)], ***and*** **Xiaoguang LIU**[†b)], ***Nonmembers***

SUMMARY With the increase of data quantity, people have begun to attach importance to cloud storage. However, numerous security accidents occurred to cloud servers recently, thus triggering the thought about the security of traditional single cloud. In other words, traditional single cloud can't ensure the privacy of users' data to a certain extent. To solve the security issues, multi-cloud systems which spread data over multiple cloud storage servers emerged. They employ a series of erasure codes and other keyless dispersal algorithms to achieve high-level security. But non-systematic codes like RS require relatively complex arithmetic, and systematic codes have relatively weaker security. In terms of keyless dispersal algorithms, they avoid key management issues but not suit to complete parallel optimization or deduplication which is important to the limited cloud storage resource. So in this paper, we design a new kind of XOR-based non-systematic erasure codes - Privacy Protecting Codes (PPC) and a SIMD encoding algorithm for better performance. To achieve higher-level security, we put forward a novel deduplication-friendly dispersal algorithm called Hash Cyclic Encryption−PPC (HCE-PPC) which can achieve complete parallelization. With these new technologies, we present a multi-cloud storage system called CloudS. For better user experiences and tradeoffs between security and performance, CloudS provides multiple levels of security by a variety of combinations of compression, encryption and coding schemes. We implement CloudS as a web application which doesn't require users to perform complicated operations on local.

key words: *multi-cloud, multi-level security, erasure code, data dispersal, key management*

1. Introduction

Cloud storage has become the focus of public attention in recent years. Due to the ever-growing amount of data and local storage limits, more and more individual and enterprise users have begun to put data in cloud servers. According to iiMedia Research Group's report [1], only in China, the amount of individual cloud storage users will reach 450 million in 2015. With the popularity of cloud storage, many Internet companies have launched their own cloud services such as Microsoft OneDrive, Amazon S3 and Google Drive. However, cloud storage security issues are gradually exposed. In 2014, photo leakage occurs to icloud which is another security incident after Google Docs and Amazon S3 [2] [3]. We summarize potential security hazards of cloud storage as the damage to data integrity and privacy. For data integrity, network transmission errors, hacker attacks and faulty operations from server administrators will cause data tampering and loss. For data privacy, both untrusted cloud providers and hacker attacks will leak out users' data. Besides, the availability of cloud services and vendor lock-in should be worth attention.

To solve the above problems, many multi-cloud systems whose main idea is data dispersal to multiple cloud servers have appeared such as DepSky [4], RACS [5], HAIL [3]. Existing multi-cloud systems typically use erasure codes (e.g. RAID-5 [6] and RS codes [7] [8]) and other keyless dispersal algorithms for data security. However, traditional erasure codes are generally systematic codes whose security is relatively weak, and non-systematic codes like multi-erasure RS require relatively complex finite field arithmetic, besides, multi-erasure seems unnecessarily expensive for only a few cloud services. In the field of keyless dispersal algorithms, traditional algorithms adopt random keys to ensure data security, which doesn't suit to deduplication. Although CAONT−RS [9] has been proposed, CAONT relies on the hash value of entire data file which is difficult for complete parallelization. So in this paper, we propose a new kind of XOR-based non-systematic erasure codes called PPC (Privacy Protecting Code) and design a SIMD encoding algorithm for better performance. In the meantime, we put forward a novel keyless dispersal algorithm called Hash Cyclic Encryption−PPC (HCE−PPC).

With various types of data uploaded to cloud servers, users' security demands for cloud systems have gradually diversified. Processing a less-confidential file with an overly complex security mechanism will lead to resource waste and bad user experiences, especially for those devices with a relatively weak computing capability. Therefore, multi-level security, which means that users can choose different security levels for their different data according to their needs, is needed to achieve trade-offs between security and performance. Unfortunately, existing multi-cloud systems don't involve this trade-off. In this paper, we design and implement a multi-cloud system called CloudS which employs different combinations of compression, encryption and coding algorithms to achieve multi-level security. As a third party agent, CloudS is allowed to temporarily access user data only when it has got tokens. For better performance and user experiences, we adopt a security authorization mechanism proposed in [10]. In addition, we implement CloudS as a web application, therefore not requiring any complicated operation even installing a software on local, reducing troubles due to changing a computer.

In summary, these are the major contributions of our

[†]College of Computer and Control Engineering & College of Software, Nankai University, China
a) E-mail: wgzwp@nbjl.nankai.edu.cn
b) E-mail: liuxg@nbjl.nankai.edu.cn
DOI: 10.1587/transinf.E0.D.1

work:

- Design a family of XOR-based non-systematic coding schemes called Privacy Protecting Code (PPC) and implement its SIMD encoding algorithm for better performance.
- Design and implement a novel keyless dispersal algorithm called HCE−PPC.
- Design and implement a variety of combinations of compression, encryption and coding schemes which provide multiple levels of trade-offs between security and performance.
- Implement the secure web-based multi-cloud storage system CloudS with the integration of the above methods.

The rest of this paper is organized as follows. In Section 2 we will discuss relevant work about multi-cloud systems and related techniques. Section 3 will introduce the framework of CloudS. Section 4 describes PPC codes in detail. Section 5 includes the design of HCE−PPC. Section 6 focuses on the security levels. In Section 7 we evaluate the performance of PPC, different security combination schemes and CloudS system. And Section 8 concludes this paper and the future work is proposed.

2. Related Work

With the prevalence of cloud storage services and broadband Internet access, data confidentiality, integrity and availability problems in traditional cloud storage raise concern gradually. To address these problems, duplication, erasure codes like MDS [11], Rabin's information dispersal [12] and Shamir's secret sharing [13] are widely used in multi-cloud storage systems which spread data over multiple cloud sites.

In the field of multi-cloud systems, RACS is a proxy that transparently spreads the storage load over many providers with RAID-like techniques to provide high data availability and avoid the costs of vendor lock-in [5]. Aimed at guaranteeing data integrity and availability, HAIL is a distributed cryptographic multi-cloud system [3] which combines proofs of retrievability (PORs) [14] and proofs of data possession (PDPs) [15]. ICStore [16] addresses CIRC (confidentiality, integrity, reliability and consistency) by using replication and Shamir's secret sharing. Depsky, a virtual storage cloud system, addresses availability and confidentiality of data by the combination of encryption, encoding and data replication [4]. Nevertheless, the above systems don't consider about various security requirements for different data and/or different users.

This paper fills that gap by presenting the systematic description of a multi-level security system called CloudS. Meanwhile, aimed at the problems of existing erasure codes and dispersal algorithms, we design a new kind of XOR-based non-systematic erasure codes called PPC which ensures data security even if one cloud server have been attacked, thus ensuring data confidentiality, integrity and fault tolerance with low storage and computational cost. We also come up with a novel dispersal algorithm named HCE−PPC which is suitable for complete parallelization and deduplication.

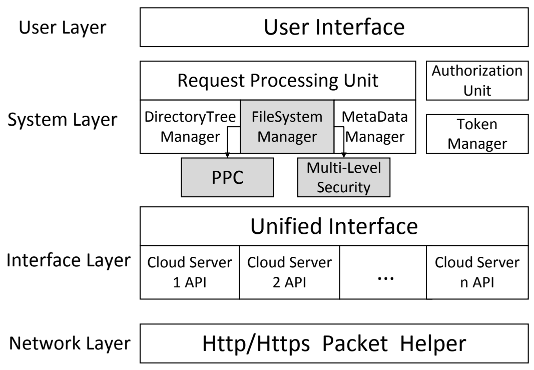


Fig. 1: CloudS architecture

3. The Framework Design

CloudS consists of four layers as presented by Fig.1. The user layer is used to accept user's requests and forward these requests to the system layer. The interface layer is designed to implements a unified interface for various cloud service providers. The network layer is responsible for sending and receiving http/https packets. The system layer is the core of CloudS which undertakes all computing tasks. It's mainly composed of the request processing unit, the authorization unit and the token manager. The authorization unit and token manager are designed to get the user's authorization tokens and hold them securely. The operation processing unit is composed of three submodules: the directorytree manager is designed for directory operation; the metadata for directories and files are manged by the metadata manager; the filesystem manager is designed to process user data and it ensures data security through a series of mechanisms. Different security levels which take full account of the user experience and achieve the trade-offs between performance and security are provided in the filesystem manager. In addition, PPC and HCE−PPC are implmented in this module as one of the coding and security schemes.

4. PPC Design and Implementation

As presented, we propose PPC to meet the tradeoffs between security and performance. In this section, we explain PPC in detail.

4.1 Nomenclature

Before introducing PPC, we first define the terms used in this section.

Chunk [17]: A basic unit of storage holding data and/or coding ('parity') information. This is also referred to as a stripe unit or a strip [18] in traditional storage systems.

Stripe: In our cloud storage model, the user file is split into k chunks and then are encoded into a stripe composed of $n(= k + m)$ chunks. These chunks are spread across n nodes to provide high reliability and availability. From the theoretical view, a stripe is a codeword which is the minimum (and complete) collection of (data and parity) bits that encode and decode together [19].

Node: An independent storage container that stores data and parity chunks. This can be a cloud storage service, a node in peer-to-peer system, and so on.

For convenience, we employ D_i to present the $(i+1)$-th original data chunk and P_i for the $(i+1)$-th coding chunk in a stripe in the following parts.

4.2 Metrics of Privacy Protecting

Privacy degree. *PPC code has a privacy degree of t, if it can resist any t breaches. That is, given any t out of n parity chunks, we cannot reconstruct any data chunk, and there exists* $(t + 1)$ *parity chunks that can reconstruct some (not necessarily all) data chunks.*

As described before, PPC code is a kind of non-systematic erasure code, so no original data is retained. In other words, we cannot reconstruct any data if we only get a single coding chunk. Hence, the privacy degree of PPC must be greater than 1. In this paper, we define $PPC(k, n, t)$ as the code in which k original data chunks generating n coding chunks and all original chunks can be recovered when holding at least k coding chunks, achieving $(n-k)$-erasure correcting and t presents privacy degree.

Safe and unsafe group. *Let S be a subset of chunks in a stripe. If we cannot reconstruct any data chunk from subset S, we call S a safe group, otherwise an unsafe group.*

We define the best case privacy degree of a PPC code as the size of its biggest safe group. Taking the PPC code in Fig. 2 as an example, we can see the chunk D_4 can be reconstructed by the XOR-sum of P_0, P_2 and P_6 and any data chunk cannot be computed by any pair of parity chunks, so the privacy degree is 2. However, not any three out of those seven parity chunks can reconstruct an original chunk. We can see that the first 5 chunks form a safe group, and any bigger group is unsafe, so its best case privacy degree is 5.

S-safe partible and s-safe partition. *A PPC code is s-safe partible if its stripe can be split into s safe groups. These groups form an s-safe partition of this PPC code.*

For a distributed storage system, attackers may breach relevant multiple sites simultaneously. In the multi-cloud environment, cloud services which come from the same country may be regarded as relevant nodes if data security involves the issue of government intervention. In other cases, natural disasters will cause all nodes within a region unavailable. The concept of safe partition offers a solution to these. For example, if we have four providers from one country and three from another country, we can employ the 2-safe partition $\{\{P_0, P_2, P_4, P_6\}, \{P_1, P_3, P_5\}\}$ of $PPC(6, 7, 2)$ shown in Fig. 2, so attackers cannot reconstruct original data even if he breaks all the cloud severs belonged to one country.

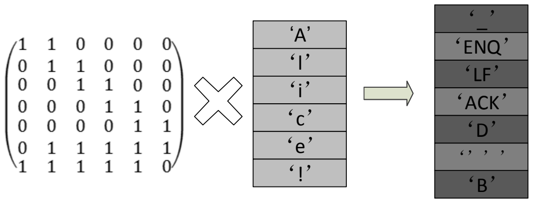


Fig. 2: PPC conversion

4.3 Structural Properties of $PPC(k, k+1, t)$

As described before, PPC is a kind of XOR-based (binary) codes, so its encoding and decoding can be achieved by multiplying the generator matrix with the vector of original data chunks. Hence, for a $PPC(k, k+1, t)$, we need a $(k+1) \times k$ generator matrix G. To tolerate any single node fault, G must have the following properties [20]:

1. Any k of G's row vectors are linearly independent. In other words, every k × k square sub-matrix of G is invertible.

2. Casually choose a row v in G, then ONES(v)≥ 2 .

The second property ensures data privacy. We define the function $ONES(v)$ as the weight of the vector v, that is, the number of ‘1’ entries in v. As discussed before, the privacy degree of PPC is greater than 1, so we only take use of those generator matrices which are purely composed of row vectors whose weights are greater than or equal to 2.

4.4 A Family of $PPC(k, k+1, \lfloor \frac{k}{2} \rfloor - 1)$

We construct an infinite family of $PPC(k, k+1, \lfloor \frac{k}{2} \rfloor - 1)$ which is named as ShrPPC. We start with constructing the first $(k-1)$ row vectors of generator matrices by shifting two consecutive set bits as shown in Formula 1. Then, we

$$\begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix} \quad (1) \qquad \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \quad (2)$$

$$\begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \quad (3) \qquad \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 1 & \cdots & 1 & 1 \\ 1 & 0 & 1 & \cdots & 1 & 0 \end{pmatrix} \quad (4)$$

construct the k-th row vector. If k is even, it is constructed as $\overrightarrow{\mathbf{01}_{k-1}}$, otherwise, we set r_k as $\overrightarrow{\mathbf{001}_{k-2}}$. Finally, we set the last row vector as the XOR-sum of the first k row vectors. When k is even, it is $\overrightarrow{\mathbf{1}_{k-1}\mathbf{0}}$. When k is odd, it is $\overrightarrow{\mathbf{101}_{k-3}\mathbf{0}}$. Formula 3 and 4 describe the generator matrices of ShrPPC with even and odd k respectively.

We can see that both matrices (when k is even or odd) satisfy the second property of generator matrix and we will prove they also satisfy property 1. First, we transform the first k rows into an upper triangular matrix (as shown in Formula 2) by performing some elementary matrix transformations. If k is even, we add the $2i$-th row to r_k for all $i \in \{1, 2, ..., \frac{k-2}{2}\}$. And if k is odd, add the $(2i+1)$-th row to r_k for all $i \in \{1, 2, ..., \frac{k-3}{2}\}$. Since the upper triangular matrix is full rank, the first k rows are linearly independent. The $(k+1)$-th row is the XOR-sum of first k rows, so any $k \times k$ sub-matrix of ShrPPC generator matrix is invertible.

ShrPPC provides a good privacy degree of $t = \lfloor \frac{k}{2} \rfloor - 1$, which is very close to the optimal privacy degree of $t = \lfloor \frac{k}{2} \rfloor$, a XOR-based $PPC(k, k+1, t)$ can achieve. For the lack of space, we omit the proof.

4.5 Searching Other Good $PPC(k, k+1, t)$

The minimum row-weight of ShrPPC's generator matrices is 2, that is, each parity chunk is generated by at least two data chunks. It provides high-level performance, but original data chunks may be stolen after the chosen-plaintext attack, if they are encoded with row vectors whose row-weight equals to 2. For example, suppose an attacker has two parity chunks, one of which is $D_0 + D_1$ and the other of which is $D_1 + D_2$. Then he knows the value of $D_0 + D_2$ as well, and it's not complex to guess unencrypted D_0, D_1, and D_2 from these information. Assuming another extreme case that D_0 is a full-zero chunk because of a special file format, it's obvious that D_1 is exactly the first parity chunk.

In view of those cases, we design a backtracking algorithm to search other good PPC by enumerating valid generator matrices with a larger minimum row-weight. The process of enumerating is time-consuming, so we use the structural properties of generator matrix to prune the search space effectively.

In the backtracking algorithm, we construct the row vectors of the generator matrix G one by one by enumerating all valid binary vectors with a weight greater than 2. Here we use "valid" means that the i-th row is linearly independent with the first $i-1$ rows. Once we successfully construct k valid row vectors for G, , we construct the last row vector as the XOR-sum of the first k row vectors and calculate the privacy degree of the new matrix. We update the optimal code if the new matrix is a better one. After that (or after an unsuccessful linear independence check), we continue to try other valid vectors and backtracking will be used if we exhausted all of the valid vectors for the current row.

Using the backtracking algorithm, we have found optimal generator matrices whose minimum row-weight is larger than (or equals to) 3 when $k < 9$ (an example is shown in Fig. 3). In addition, we can resist the chosen-plaintext attack by means of the combination of PPC and compression and/or encryption.

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Fig. 3: An example of generator matrix with a minimum row-weight of 3

4.6 Optimal Schedule of PPC Encoding

PPC is a kind of XOR-based code that each coding chunk P is decided by the corresponding row v in the generator matrix G. The more '1' v has, the more XOR operations performed in computing P. Suppose G has two vectors of $v_i = \overrightarrow{\mathbf{001}_{k-2}}$ and $v_j = \overrightarrow{\mathbf{0001}_{k-3}}$ and therefore it takes $(2k-7)$ XOR operations to calculate the two parity chunks P_i and P_j. However, if we calculate P_j first, and then calculate P_i by XORing P_j and D_2, only $(k-3)$ XOR operations are needed. Since this paper focuses on small-scale multi-cloud systems, we adopt a breadth-first search algorithm [21] to determine the optimal schedule for PPC encoding.

5. HCE−PPC Dispersal Algorithm

In the study of multi-cloud, keyless dispersal algorithms are widely used to ensure data security and fault-tolerant. They avoid key management issues but not suit to complete parallel optimization or deduplication. So in this section, we propose the HCE−PPC dispersal algorithm which is easy to be parallelized and is deduplication-friendly as well. We also analyze the security of HCE−PPC and its application in multi-cloud environment.

5.1 The Algorithm of HCE−PPC

In HCE−PPC, we perform PPC after a Hash Cyclic Encryption (HCE). We first divide the original data into several chunks and encrypt these chunks with their own hash values, therefore bringing high concurrency. Due to the "chunk-level hash key", HCE can be also convenient for deduplication in cloud environment, which can remit the vital storage space. For separating cipher and secret keys, hash values (secret keys) are stored circularly, and that method of key-storage can help HCE−PPC to achieve nearly the same security as AONT−RS.

As presented, PPC is a family of single-fault-tolerance codes, so if we divide the original data into n chunks, $(n+1)$ chunks are generated after HCE−PPC. Hereinafter, we take $ShrPPC(6, 7, 2)$(shown in Fig.2) for example. Fig.4 shows the idea of HCE−PPC. Here we use D_i to denote the $(i+1)$-th original data chunk, C_i means the $(i+1)$-th cipher chunk

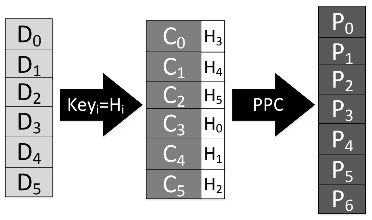


Fig. 4: HCE−PPC

and P_i signifies the $(i + 1)$-th coding chunk after PPC. In addition, Key_i presents the key used in encrypting D_i and H_i is the hash value of D_i. The steps of HCE−PPC are as follow:

Step1: Calculate the hash value H_i of D_i by

$$H_i = \mathbf{H}(D_i), for\ i = 0, 1, \cdots, 5. \quad (5)$$

Here, **H** is a common hash function (e.g. MD5, SHA-256).

Step2: Encrypt D_i into C_i by

$$C_i = \mathbf{E}(H_i, D_i), for\ i = 0, 1, \cdots, 5. \quad (6)$$

E is an encryption function (e.g. AES-256). In this step, we encrypt each original data chunk using its own hash value. We know that only the same data chunk will obtain the same hash value via the same hash function. Due to the uniqueness of different hash value, the same original chunk leads to the same cipher chunk in HCE, which is benefit for deduplication. Besides, HCE no longer depends on the whole cipher's hash value, which is better for parallelization.

Step3: Append H_i at the end of $C_{(i+3)\text{mod}6}$.
This circular key placement enhances the security of HCE−PPC, we will detailedly analyze it in the following part. In this step, we gain the cipher mode as $\{C_i, H_{(i-3)\text{mod}6}\}$.

Step4: Perform PPC on those $\{C_i, H_{(i-3)\text{mod}6}\}$.

5.2 The Security of HCE−PPC

We mainly owe the security of HCE−PPC to the confidentiality of encryption (e.g.AES-256) and hash functions, key storage and the characteristics of PPC arithmetic. As described, PPC is a family of non-systematic codes, there're no original data retaining in coding data. So in HCE−PPC, PPC itself can play a protective role for data to some extent. For $PPC(k, k + 1, t)$, any m ($m \leq k - 1$) chunks cannot retrieve the whole uncoded data, any p ($p \leq t$) chunks cannot break out any original data.

It's no doubt that decrypting a message requires both key and encrypted data, that is, to decrypt arbitrary D_i, we need obtain both C_i and H_i which are stored in the $(i+1)-th$ and the $((i + 4)\text{mod}6)$ − th uncoded chunks. From the generator matrix of PPC (as Fig.2), we can discover that decoding the first orignal chunk needs P_1, P_3 and P_6, the second chunk needs P_2, P_4 and P_5, the third needs P_0, P_3 and P_6, the fourth needs P_1, P_4 and P_5, the fifth needs P_0, P_2 and P_6 and P_1, P_3, P_5 for the last chunk. It's obvious that retrieving both the $(i + 1) - th$ and the $((i + 4)\text{mod}6)$ − th chunks needs 5 different coding chunks, therefore realizing the approximate security compared by AONT−RS. As presented, PPC code has the property of 2-safe partition, we can divide the encoded chunk in two safe groups: $\{P_0, P_2, P_4, P_6\}$, $\{P_1, P_3, P_5\}$, that's to say, decoding C_i and H_i needs chunks from both two safe groups, achieving the cross protection of secret key which also solves the government intervention.

Besides, as intrdoduced, PPC may come under the chosen-plaintext attack, HCE−PPC is a better precept to avoid it. We perform PPC after HCE, so HCE−PPC avoids the possible chosen-plaintext attack because we perform PPC on cipher instead of plaintext.

5.3 HCE−PPC in Multi-cloud Environment

From the above analysis, it's not difficult to conclude that HCE−PPC realizes its security by the separation of data storage. That is, when different data chunks are stored in isolated physical region, HCE−PPC can really play the role. Besides, PPC is binary codes and support single-fault-tolerance, therefore, HCE−PPC indeed fits the reliability and availability requirements of multi-cloud environment. With HCE−PPC, coding chunks are randomly spread to different cloud sites. As described, 5 coding chunks are required for an original plaintext chunk, in other words, attackers needs to break 5 cloud servers in s short time (it's nearly impossible). On account of the high-level security of HCE−PPC, we employ it as one of the security schemes in CloudS to provide users with multi-level security.

6. Multi-Level Security

For better user experiences, CloudS provides some security levels with different characteristics. From the system perspective, each level logically corresponds to a specific combination scheme of compression, encryption and coding. In CloudS, we implement not only combinations of some existing technologies such as RS, RAID-5, AES [22] and AONT, but some new technologies such as PPC and HCE−PPC. In this section, we list three new combination schemes to introduce different security levels in the system view.

6.1 HCE−PPC

HCE−PPC is regarded as one of the security levels. Due to the detailed presentation in the above section, we only focus on the applicable scenario of HCE−PPC in this part. Compared with the following schemes containing compression, HCE−PPC provides stable performance without the influence of the compression ratio but more data will be transferred on Internet, thereby satisfying the small-size files with higher compression ratio and high security requirements.

6.2 Compression−PPC and Compression−HCE−PPC

Considering the time-consuming network transmission process, GZip [23] compression is joint in these two schemes. Compared with AONT−RS, these two schemes have an overwhelming advantage in network transmission. As for security, non-systematic $PPC(k, k+1, t)$ itself has an effect of data protection, t (greater than 1) coding chunks (means at least two cloud servers) are needed for retrieving only one original chunk, which is very difficult. Meanwhile, the chosen-plaintext attack will be remitted since original data patterns will be destroyed after compression. Nonetheless, decompression doesn't need any key, so Compression−PPC has relatively weak security but high speed. We suggest users to adopt Compression−PPC for less-important data. Compression−HCE−PPC is the synthesis of HCE−PPC and Compression−PPC which meets the demands for large confidential data.

7. Evaluation

In this paper, we designed a family of non-systematic erasure codes named PPC and proposed a novel dispersal algorithm called HCE−PPC. Meanwhile, on the basis of the above technologies, we implemented a multi-level multi-cloud system, CloudS, to ensure data security in cloud storage environment. In this section, we will test their performance. Our prototype is implemented in C++, the experiments are conducted on Ubuntu 13.04 with Intel i3-350M@2.26GHz.

7.1 PPC Performance

There're two important indicators to measure PPC, the encoding and decoding speed. To measure those metrics, we implement encoders and decoders for PPC whose generator matrix is shown in Fig. 3. As the control group we implement RS (we only consider about the capability of reconstruction in multi-cloud environments). For better performance, we also realize SIMD (SSE) optimization on PPC and optimized PPC (mentioned in §4.6). Fig. 5 and 6 present their encoding and decoding speeds. For PPC, all the operations are over GF(2), it's obvious that XOR operation has a higher-speed than operations over GF(2^8) used by RS. In the same time, it's worth attention that SSE optimization has indeed a great performance improvement (around 6X∼7X). In the following part, all experiments are conducted using PPC−SSE. Besides, we compare encoding time between the original PPC scheme and the optimized PPC schedule. The former needs 15 XOR operations to encode a stripe while the latter needs only 11 XOR operations. It's clear that the optimization becomes more obvious when encoding larger data.

PPC enables cloud storage customers to explore trade-offs between a little extra storage cost, data confidentiality and integrity. Meanwhile, SSE optimization leads to a prominent performance improvement. Therefore, we believe that PPC is practical choice in the multiple clouds environment.

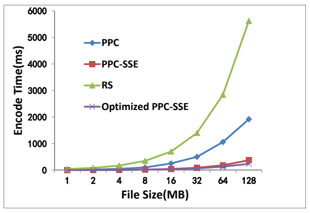


Fig. 5: PPC encoding performance

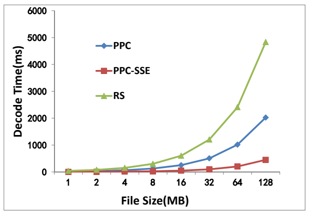


Fig. 6: PPC decoding performance

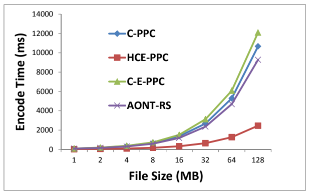


Fig. 7: Encoding speed of different security schemes

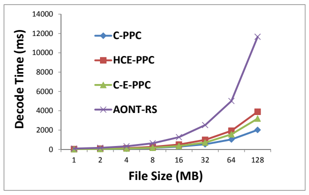


Fig. 8: Decoding speed of different security schemes

7.2 Performance of Different Security Schemes

In this part, we report scheme performance without network transmission process. We use C−PPC to represent Compression−PPC and C−E−PPC to symbolize the Compression−HCE−PPC. Fig.7 presents the encoding speed, two schemes without compression do better because the compression speed is much lower than encryption. HCE−PPC has the highest performance due to PPC's better capability. Fig.8 distinguishes the decoding speed of different schemes. Unlike slow compression, decompression is much faster, therefore, schemes with compression perform better. We can clearly figure that the performance gap of schemes is widening when the data amount increases. Above all, whether for encoding or decoding performance, HCE−PPC indeed excels AONT−RS.

7.3 System Performance

In CloudS, there're three steps in uploading and downloading process: the transmission between client and CloudS agent, operations on CloudS and the transmission between CloudS and cloud servers. The performance of the first step is up to network condition which is out of our research range, so we only evaluate the last two steps. It's notable that some uncontrolled exterior factors such as network condition and the compression ratio of files will also influence our results. We create accounts on Vdisk to simulate multiple clouds environment and adopt randomly generated test files. We run Apache on PC for test which means a real web server will do better. Table1 and 2 respectively illustrate the uploading and downloading performance. We can see that the schemes containing compression do much better due to less data being delivered. C−PPC does the best on account

Table 1: System Uploading Performance(s)

File Size(MB)	1	8	16	64	128
C−PPC	0.87	3.90	7.29	21.01	46.28
HCE−PPC	1.16	5.59	13.63	51.02	98.95
C−E−PPC	1.07	4.86	8.12	23.41	50.03
AONT−RS	1.46	8.06	17.37	56.55	108.98

Table 2: System Downloading Performance(s)

File Size(MB)	1	8	16	64	128
C−PPC	0.98	4.25	9.40	27.38	52.37
HCE−PPC	1.66	7.44	15.39	51.09	103.94
C−E−PPC	1.21	5.42	10.35	29.98	56.63
AONT−RS	1.81	8.78	18.94	57.13	114.02

of less data to be transferred and less mechanism being used. HCE−PPC indeed has better performance than AONT−RS. Certainly, all those schemes are in the range users can accept.

8. Conclusion and Future Work

With the development of cloud storage, the security issues become more and more serious, data integrity and availability is being threatened. We design a new family of erasure codes called PPC to guarantee data security. PPC is XOR-based codes that results in a low storage computational cost, especially after parallel optimization. Simultaneously, as a single-fault-tolerance code, PPC$(k, k+1, t)$ only need $\frac{(k+1)}{k}$ times extra storage overhead to achieve the trade-offs between storage and fault tolerance. It is important to note that PPC, as a kind of non-systematic code, has a better function of data confusion. In the field of dispersal algorithm, we propose a novel deduplication-friendly HCE−PPC dispersal algorithm which can achieve complete parallelization.

With PPC and HCE−PPC algorithm, we put forward a multi-cloud system called CloudS. Though many multi-cloud systems have appeared in recent years such as RACS and Depsky, they cannot fully adapt to users' different requirements for different files in cloud environment. The single-level security scheme will give rise to a worse user experience and system performance. Redundant and complex security mechanisms will become the bottleneck of the system, particularly for the devices with a weak computing ability and limited computing resources. In CloudS, we provide several different combination schemes of compression, encryption and coding to achieve multi-level security for better performance and user experiences. At the same time, we implement CloudS as a web application which doesn't require users to do more operation locally.

In our future work, we would like to extend our PPC codes to resist double fault. In addition, we can still optimize CloudS in terms of performance.

Acknowledgments

This work is partially supported by NSF of China (grant numbers: 61373018, 11301288), Program for New Century Excellent Talents in University (grant number: NCET130301) and the Fundamental Research Funds for the Central Universities (grant number: 65141021).

References

[1] "China personal cloud storage industry and users' behavior research." `http://www.iimedia.cn/38351.html`. Accessed December 29, 2014.

[2] C. Cachin, I. Keidar, and A. Shraer, "Trusting the cloud," Acm Sigact News, vol.40, no.2, pp.81–86, 2009.

[3] K.D. Bowers, A. Juels, and A. Oprea, "Hail: a high-availability and integrity layer for cloud storage," Proceedings of the 16th ACM conference on Computer and communications security, pp.187–198, ACM, 2009.

[4] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," ACM Transactions on Storage (TOS), vol.9, no.4, p.12, 2013.

[5] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "Racs: a case for cloud storage diversity," Proceedings of the 1st ACM symposium on Cloud computing, pp.229–240, ACM, 2010.

[6] D.A. Patterson, G. Gibson, and R.H. Katz, A case for redundant arrays of inexpensive disks (RAID), ACM, 1988.

[7] K. Rashmi, P. Nakkiran, J. Wang, N.B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for i/o, storage, and network-bandwidth," Proceedings of the 13th USENIX Conference on File and Storage Technologies, pp.81–94, USENIX Association, 2015.

[8] J.S. Plank *et al.*, "A tutorial on reed-solomon coding for fault-tolerance in raid-like systems," Softw., Pract. Exper., vol.27, no.9, pp.995–1012, 1997.

[9] M. Li, c. Qin, P.P. Lee, and J. Li, "Convergent dispersal: Toward storage-efficient security in a cloud-of-clouds," Proceedings of the 6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage' 14), pp.1–5.

[10] J. SUN, M. XU, S. FENG, Z. LI, G. WANG, and X. LIU, "Secure store of user authentication tokens in multi-cloud storage system," Journal of Computational Information Systems, vol.11, no.3, pp.1013–1020, 2015.

[11] R.Singleton, "Maximum distance-nary codes," IEEE Transactions on Information Theory, pp.116–118, IEEE, 1964.

[12] M.O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," Journal of the ACM (JACM), vol.36, no.2, pp.335–348, 1989.

[13] A. Shamir, "How to share a secret," Communications of the ACM, vol.22, no.11, pp.612–613, 1979.

[14] A. Juels and B.S. Kaliski Jr, "Pors: Proofs of retrievability for large files," Proceedings of the 14th ACM conference on Computer and communications security, pp.584–597, Acm, 2007.

[15] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," Proceedings of the 14th ACM conference on Computer and communications security, pp.598–609, Acm, 2007.

[16] C. Cachin, R. Haas, and M. Vukolic, "Dependable storage in the intercloud," tech. rep., Research Report RZ, 3783, 2010.

[17] S. Ghemawat, H. Gobioff, and S.T. Leung, "The google file system," ACM SIGOPS operating systems review, pp.29–43, ACM, 2003.

[18] J.S. Jason K.Resch, "Aont-rs:blending security and performance in dispersed storage systems.," FAST, 2011.

[19] J. Plank and C. Huang, "Tutorial: Erasure coding for storage applications," Slides presented at FAST-2013: 11th Usenix Conference on File and Storage Technologies, 2013.

[20] J.S. Plank, M. Blaum, and J.L. Hafner, "Sd codes: erasure codes designed for how storage systems really fail.," FAST, pp.95–104, 2013.

[21] J.S. Plank, C.D. Schuman, and B.D. Robison, "Heuristics for optimizing matrix-based erasure codes for fault-tolerant storage systems," Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems & Networks, pp.1–12, IEEE/IFIP, 2012.

[22] F.P. Miller, A.F. Vandome, and J. McBrewster, "Advanced encryption standard," 2009.
[23] "Gzip." `http://en.wikipedia.org/wiki/Gzip`. Accessed April 12, 2015.

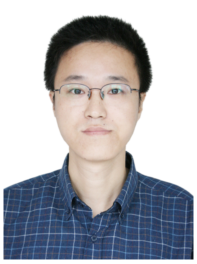
Lu Shen received her B.Sc. degree from Jilin University, Changchun, China, in 2010. Currently, she is the student of Nankai-Baidu Joint Lab, Nankai University, China. Her research interests include storage security, cloud storage, the method of encoding and encryption.

Shifang Feng received her B.Sc. degree from Nankai University, Tianjin, China, in 2010. Currently, she is the student of Nankai-Baidu Joint Lab, Nankai University, China. Her research interests include data dispersal, cloud s-torage, secure compress.

Jinjin Sun received her B.Sc. degree from Nankai University, Tianjin, China, in 2010. Currently, she is the student of Nankai-Baidu Joint Lab, Nankai University, China. Her research interests include storage secure, secure compress, token management.

Zhongwei Li received his Ph.D. degree in Computer application technology from Harbin Engineering University, Harbin, China, in 2006. He is currently an associate professor in computer science at Nankai University,Tianjin, China. His research interests include storage system and data mining

Ming Su received his B.Sc. and Ph.D. degrees in college of mathematics from Nankai University, Tianjin, China, in 1999 and 2004, respectively. He stayed in Austrian Academy of Science(Osterreichische Academie der Wissenschaften) as an OEAD scholar in 2009. His research interests include complexity of sequences, information theory, and digital right management



Gang Wang received his B.Sc., M.Sc. and Ph.D. degrees in computer science from Nankai University, Tianjin, China, in 1996, 1999 and 2002, respectively. He is currently a professor at the College of Computer and Control Engineering, Nankai University. His research interests include storage systems and parallel computing.

Xiaoguang Liu received the B.Sc. degree, M.Sc. degree and Ph.D. degree in computer science from Nankai University, Tianjin, China, in 1996, 1999 and 2002 respectively. He is currently a professor in computer science at Nankai University,Tianjin, China. His research interests include parallel computing, storage system and search engine.