Security and Efficiency Trade-offs for Cloud Computing and Storage

Jian Li, Kai Zhou and Jian Ren
Department of ECE, Michigan State University, East Lansing, MI 48824-1226
Email: {lijian6, zhoukai, renjian}@msu.edu

***Abstract*—Cloud computing provides centralized data storage and online computing. For centralized data storage, to address both security and availability issue, distributed storage is an appealing option in that it can ensure file security without requiring data encryption. Instead of storing a file and its replications on multiple servers, we can break the file into components and store the components on multiple servers. In this paper, we develop the minimum storage regeneration (MSR) point based distributed storage schemes that can achieve optimal storage efficiency and storage capacity. Moreover, our scheme can detect and correct malicious nodes with much higher error correction efficiency. For online computing, we investigate outsourcing of general computational problems and propose efficient Cost-Aware Secure Outsourcing (CASO) schemes for problem transformation and outsourcing so that the cloud is unable to learn any key information from the transformed problem. We also propose a verification scheme to ensure that the end-users will always receive a valid solution from the cloud. Our extensive complexity and security analysis show that our proposed Cost-Aware Secure Outsourcing (CASO) scheme is both practical and effective.**

***Index Terms*—Cloud computing, distributed storage, optimal scheme design, computation outsourcing, security, efficiency, cost-aware**

I. Introduction

Cloud computing paradigm provides with end-users an on-demand access to a shared pool of resources. Computational resource and storage are two of the most integral services of cloud computing. Distributed storage is an appealing method to store files securely without requiring data encryption. Instead of storing a file and its replications on multiple servers, we can break the file into components and store the components on multiple servers. In this way, both reliability and security of the file can be increased. A typical approach is to encode the file using an (n, k) Reed-Solomon (RS) code and distribute the encoded file into n servers. When we need to recover the file, we only need to collect the encoded parts from k servers, which achieves a trade-off between reliability and efficiency. However, when repairing or regenerating the contents of a failed node, the whole file has to be recovered first, which is a waste of bandwidth. The concept of regenerating code was introduced in [1]. The bandwidth needed for repairing a failed node could be much less than the size of the whole file. In fact the regenerating code achieves an optimal trade-off between bandwidth and storage within the minimum storage regeneration (MSR) point.

In the recent years, the security of the regenerating code has been studied. In [2], the authors analyzed the error resilience of the regenerating code in the network with errors and erasures. They provided the theoretical error correction capability. In [3] the authors proposed a Hermitian code [4] based regenerating code, which could provide better error correction capability. In [5] the authors proposed the universally secure regenerating code to achieve information theoretic data confidentiality. But the extra computational cost and bandwidth have to be considered for this code. In [6] the authors proposed to apply linear feedback shift register (LFSR) to protect the data confidentiality.

Computation outsourcing is another key component of cloud computing. It enables the resource-constrained end-users to outsource their computational tasks to the cloud servers so that the tasks can be processed in the cloud servers. However, security has become one of the major concerns that prevent computation outsourcing from being widely adopted. When the end-users outsource their tasks to the cloud, the cloud servers will get full access to the problem. As a result, the end-users' privacy is totally exposed to the cloud. Furthermore, the cloud may have the motivation to cheat in the computation process thus false solutions may be returned to the end-users. All these issues call for designs of secure outsourcing mechanisms that can also provide end-users the ability to validate the received results.

In our research, we develop the minimum storage regeneration (MSR) and the minimum bandwidth regeneration (MBR) points based distributed storage schemes that can achieve *optimal* storage efficiency and storage capacity. Our scheme can detect and correct malicious nodes with much higher error correction capability at a much lower computation efficiency. For computation outsourcing, we propose cost-aware secure outsourcing (CASO) schemes based on affine mappings for problem transformation and solution recovery. We also propose efficient result verification schemes.

The rest of the paper is organized as follows: In Section II, we present our proposed optimal distributed storage. In Section III, our proposed cost-aware secure computation outsourcing scheme is presented. We conclude in Section IV.

II. Optimal Distributed Storage

Distributed storage provides an efficient and low-cost trade-off between security and storage. It changes the costly security management problems to storage problems. Meanwhile, it also provide an elegant solution to ensure data availability under various security attacks and natural disaster. The idea of distributed storage is to encode the file using an (n, k) Reed-Solomon (RS) code and distribute the encoded file into n servers. When we need to recover the file, we only need to collect the encoded parts from k servers, which achieves a trade-off between reliability and efficiency. However, when repairing or regenerating the contents of a failed node, the whole file has to be recovered first, which is a waste of bandwidth. The concept of regenerating code was introduced in [1]. The bandwidth needed for repairing a failed node could be much less than the size of the whole file. In fact the regenerating code achieves an optimal trade-off between bandwidth and storage within the minimum storage regeneration (MSR) and the minimum bandwidth regeneration (MBR) points.

A. Preliminary

1) Regenerating Code: Regenerating code introduced in [1] is a linear code over finite filed $\mathbb{F}_q$ with a set of parameters $\{n, k, d, \alpha, \beta, B\}$. A file of size B is stored in n storage nodes, each of which stores α symbols. A replacement node can regenerate the contents of a failed node by downloading β symbols from each of d randomly selected storage nodes. So the total bandwidth needed to regenerate a failed node is $\gamma = d\beta$. The data collector (DC) can reconstruct the whole file by downloading α symbols from each of $k \leq d$ randomly selected storage nodes [1], the following theoretical bound was derived:

$$B \leq \sum_{i=0}^{k-1} \min\{\alpha, (d-i)\beta\}. \tag{1}$$

From Equation (1), a trade-off between the regeneration bandwidth γ and the storage requirement α was derived. γ and α cannot be decreased at the same time. There are two special cases: minimum storage regeneration (MSR) point in which the storage parameter α is minimized;

$$(\alpha_{MSR}, \gamma_{MSR}) = \left(\frac{B}{k}, \frac{Bd}{k(d-k+1)}\right), \tag{2}$$

and minimum bandwidth regeneration (MBR) point in which the bandwidth γ is minimized:

$$(\alpha_{MBR}, \gamma_{MBR}) = \left(\frac{2Bd}{2kd-k^2+k}, \frac{2Bd}{2kd-k^2+k}\right). \tag{3}$$

2) Encoding of MSR Code: The encoding is based on the product-matrix code framework [7]. According to Equation (2), we have $\alpha_H = d/2$, $\beta_H = 1$ for one block of data with the size $B_H = (\alpha+1)\alpha$. The data will be arranged into two $\alpha \times \alpha$ symmetric matrices S_1, S_2, each of which will contain $B_H/2$ data. The codeword CH is defined as

$$\mathsf{CH} = [\Phi \quad \Lambda\Phi] \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \Psi S_H, \tag{4}$$

where

$$\Phi = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \phi & \phi^2 & \dots & \phi^{\alpha-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \phi^{n-1} & (\phi^{n-1})^2 & \dots & (\phi^{n-1})^{\alpha-1} \end{bmatrix} \tag{5}$$

is a Vandermonde matrix and $\Lambda = \text{diag}[\lambda_1, \lambda_2, \cdots, \lambda_\alpha]$ such that $\lambda_i \neq \lambda_j$ for $1 \leq i, j \leq \alpha, i \neq j$, where $\lambda_i \in \mathbb{F}_q$ for $1 \leq i \leq \alpha$, ϕ is a primitive element in $\mathbb{F}_q$, and any d rows of Ψ are linearly independent. Then each row ch_i, $0 \leq i < n$, of the codeword matrix ch will be stored in storage node i, in which the encoding vector ν_i is the i^{th} row of Ψ.

B. m-Layer Rate-matched MSR Code

In this section, we will show our optimization of the rate-matched [8] MSR code: m-layer rate-matched MSR code. We utilize m layers of the MSR codes, with each layer L_i viewed as an $(n-1, d_i, n-d_i)$ MDS code, $1 \leq i \leq m$, satisfying

$$d_i \leq d_j, \ \forall 1 \leq i \leq j \leq m. \tag{6}$$

The data will be divided into m parts and each part will be encoded by a distinct rate MSR code. The code with a lower code rate has better error correction capability. The codewords will be decoded layer by layer in the order from layer L_1 to layer L_m. That is, the codewords encoded by the MSR code with a lower d will be decoded prior to those encoded by the full rate MSR code with a higher d. If errors were found by the full rate MSR code with a lower d, the corresponding nodes would be marked as malicious. The symbols sent from these nodes would be treated as erasures in the subsequent decoding of the MSR codes with higher d's. The purpose of this arrangement is to correct as many as erroneous symbols sent by malicious nodes and locate the corresponding malicious nodes using the MSR code with a lower rate. However, the rates of the m MSR codes must match to achieve an optimal performance. Here we mainly focus on the rate match for data regeneration. We can see in the later analysis that the performance of data reconstruction can also be improved with this design criterion.

To optimize the overall error correction capability by matching the code rates of different rate MSR codes, we set the summation of the d's of all the layers to a constant $\bar{d}$:

$$\sum_{i=1}^{m} d_i = \bar{d}. \tag{7}$$

The first layer code can correct t_1 errors. If we view the symbols from the t_{i-1} nodes where errors are found by

CH_{i-1} as erasures, the i^{th} layer code can correct t_i errors for $2 \le i \le m$:

$$\begin{aligned} t_i &= \lfloor (n - d_i - 1 - t_{i-1})/2 \rfloor + t_{i-1} \\ &= \left(\sum_{j=1}^{i} 2^{j-1}(n - d_j) - \sum_{j=1}^{i} 2^{j-1}\varepsilon_j - 2^i + 1 \right) / 2^i, \end{aligned} \tag{8}$$

where $\varepsilon_i = 0$ or 1, with the restriction that $n - d_i - 1 \ge t_{i-1}$, which can be written as:

$$-\sum_{j=1}^{i-1} 2^{j-1} d_j + 2^{i-1} d_i \le n + \sum_{j=1}^{i-1} 2^{j-1}\varepsilon_j - 1. \tag{9}$$

Similarly, the parameter d of the i^{th} layer for $2 \le i \le m$ must satisfy

$$d_{i-1} - d_i \le 0. \tag{10}$$

And Equation (7) can be written as:

$$\sum_{j=1}^{m} d_j \le \bar{d}, \tag{11}$$

$$-\sum_{j=1}^{m} d_j \le -\bar{d}. \tag{12}$$

We can maximize the error correction capability of the m-layer rate-matched MSR code by maximizing t_m. With all the constrains listed above, we can write the optimization problem as:

$$\begin{aligned} &\text{maximize} \quad t_i \text{ for } i = m \text{ in } (8), \\ &\text{subject to} \quad (9) \text{ and } (10) \text{ for } 2 \le i \le m, \ (11), \ (12). \end{aligned}$$

The optimization result is:

Theorem 1. *For the m-layer rate-matched MSR code, when*

$$d_i = \mathit{Round}(\bar{d}/m) = \widetilde{d} \quad \text{for } 1 \le i \le m, \tag{13}$$

it can correct errors from at most

$$\begin{aligned} \widetilde{t}_m &= ((2^m - 1)(n - \widetilde{d}) - \sum_{j=1}^{m} 2^{j-1}\varepsilon_j - 2^m + 1)/2^m \\ &\ge ((2^m - 1)(n - \widetilde{d}) - 2^{m+1} + 2)/2^m. \end{aligned} \tag{14}$$

malicious nodes.

The error correction efficiency for the m-layer rate-matched MSR code is

$$\delta_C = ((2^m - 1)(n - \widetilde{d}) - 2^{m+1} + 2)/(2^m n). \tag{15}$$

This is a monotonically increasing function for m, so we have:

Corollary 1. *The error correction efficiency of the m-layer rate-matched MSR code increases with the number of layers.*

Evaluation of the Optimization: Equation (13) shows that we can get the optimized error correction capability when all the rates of the codes in the m-layer code are equal. Since we have seen the advantage of the rate-matched MSR code over the normal error correction MSR code, here we will mainly discuss how the number of layers can affect the error correction efficiency. The error correction efficiency of the m-layer rate-matched MSR code is shown is Fig. 1, where we set $n = 30$ and $\bar{d} = 50$. We also plot the error correction efficiency δ'_C of the normal error correction MSR code under the same parameters for comparison. From the figure we can see that when n and $\bar{d}$ are fixed, the optimal error correction efficiency will increase with the number of layers m as in Corollary 1. Moreover, the optimization condition in Equation (13) also leads to maximum storage capacity besides the optimal error correction capability. We have the following theorem:

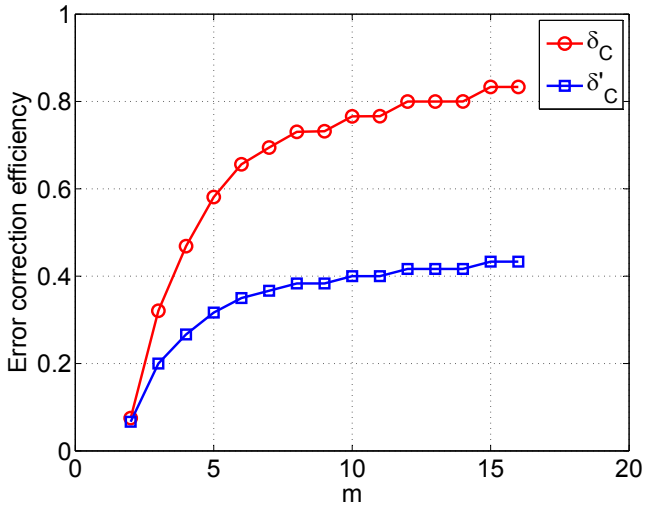

Figure 1. The optimal error correction efficiency of the m-layer rate-matched MSR code under different m for $2 \le m \le 16$

Theorem 2. *The m-layer rate-matched MSR code can achieve the maximum storage capacity if the parameter d's of all the layers are the same, under the constraint in Equation (7).*

C. Practical Consideration of the Optimization

So far, we implicitly presumed that there is only one data block of the size $B_i = \alpha_i(\alpha_i + 1)$ for each layer i. In practical distributed storage, it is the parameter d_i that is fixed instead of d_0, the summation of d_i. However, as long as we use m layers of MSR codes with the same parameter $d = \widetilde{d}$, we will still get the optimal solution for $\bar{d} = m\widetilde{d}$. In fact, the m-layer rate-matched MSR code here becomes a single MSR code with parameter $d = \widetilde{d}$ and m data blocks. And based on the dependent decoding idea we describe at the beginning of Section II-B, we can achieve the optimal performance.

So when the file size B_F is larger than one data block size $\widetilde{B}$ of the single full rate MSR code with parameter $d = \widetilde{d}$, we will divide the file into $\lceil B_F/\widetilde{B} \rceil$ data blocks and encode them separately. If we decode these data blocks dependently, we can get the optimal error correction efficiency.

Evaluation of the Optimal Error Correction Efficiency: In the practical case, $\widetilde{d}$ in Equation (15) is fixed. So here we will study the relationship between the number of dependently decoding data blocks m and the error correction efficiency δ_C,

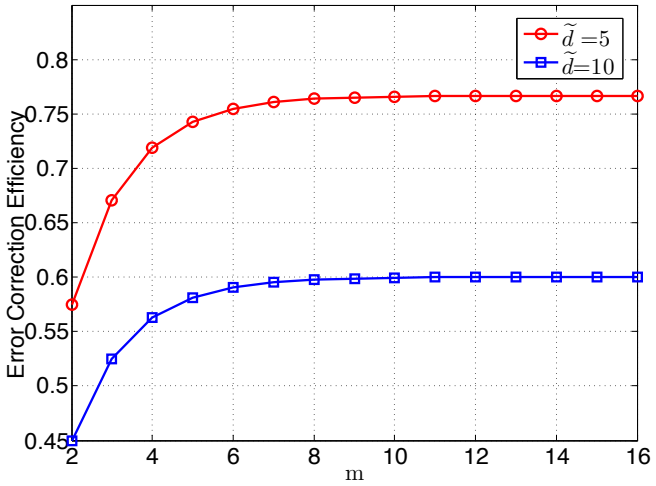

Figure 2. The optimal error correction efficiency for $2 \leq m \leq 16$

which is shown in Fig. 2. We set $n = 30$ and $\widetilde{d} = 5, 10$. From the figure we can see that although δ_C will become higher with the increasing of dependently decoding data blocks m, the efficiency improvement will be negligible for $m \geq 8$. Actually when $m = 7$ the efficiency has already become 99% of the upper bound of δ_C.

On the other hand, there exist parallel algorithms for fast MDS code decoding [9]. We can decode blocks of MDS codewords parallel in a pipeline fashion to accelerate the overall decoding speed. The more blocks of codewords we decode parallel, the faster we will finish the whole decoding process. For large files that could be divided into a large amount of data blocks (θ blocks), we can get a trade-off between the optimal error correction efficiency and the decoding speed by setting the number of dependently decoding data blocks m and the number of parallel decoding data blocks ρ under the constraint $\theta = m\rho$.

D. Encoding of m-Layer Rate-matched MSR Code

From the analysis above we know that to encode a file with size B_F using the optimal m-layer rate-matched MSR code is to encode the file using a full rate MSR code with predetermined parameter $d = 2\alpha = \widetilde{d}$. First the file will be divided into θ blocks of data with size $\widetilde{B}$, where $\theta = \lceil B_F/\widetilde{B} \rceil$. Then the θ blocks of data will be encoded into code matrices $\mathsf{CH}_1, \cdots, \mathsf{CH}_\theta$ and form the final $n \times \alpha\theta$ codeword matrix: $CM = [\mathsf{CH}_1, \cdots, \mathsf{CH}_\theta]$. Each row $\mathbf{c}_i = [\mathbf{ch}_{\mathbf{1},i}, \cdots, \mathbf{ch}_{\theta,i}]$, $0 \leq i \leq n-1$, of the codeword matrix CM will be stored in storage node i, where $\mathbf{ch}_{\mathbf{j},i}$ is the i^{th} row of CH_j, $1 \leq j \leq \theta$. The encoding vector ν_i for storage node i is the i^{th} row of Ψ in Equation (4).

For this encoding, we can construct the optimal data regeneration and reconstruction algorithms, both at complexity $\mathcal{O}(n^{5/3})$, which is lower than the complexity $\mathcal{O}(n^2)$ for RS code based regeneration code.

III. Cost-Aware Secure Computing Outsourcing

Suppose an end-user wishes to solve a computational problem denoted by $F(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ is the input sequence. Note that $F(\mathbf{x})$ describes a general computational problem not necessarily restricted to a function. For example, it can be a system of equations or an optimization problem. However, due to the lack of resources, the end-user may need to outsource the problem to the cloud which is considered to have infinite computing resources. The main idea is that before outsourcing, the end-user will transform the original problem at local side in order to conceal information leakage of the original computational problem. On receiving the transformed problem, the cloud servers will carry out the computing process and return the solution to the end-user. Based on the transformation and the information returned by the cloud, the end-user is able to recover the solution of the original problem and validate the received solution.

A. Design Requirements and Goals

A secure outsourcing scheme should satisfy the following requirements: (i) *Correctness*: the transformation on the problem and the inverse transformation should guarantee that the recovered solution is correct, (ii) *Verifiability*: the end-user has the ability to verify the validity of the solution returned by the cloud. (iii) *Secrecy*: the outsourcing scheme should prevent the cloud from learning any key information, such as the problem itself, the input and the output of the problem, (iv) *Efficiency*: The computational overhead for the outsourcing scheme and the result verification should be limited to $\mathcal{O}(n^2)$ while maintaining the original level of communication overhead, and (v) *Cost-Awareness:* The end-users can determine the appropriate outsourcing strategies according to their own computational constraints and security demands in a cost-aware manner.

B. Basic Framework

We formally divide the outsourcing process into the following phases.

1) **Problem Transformation**: $\mathsf{ProbTran}$ $\{\mathbf{S}, F(\mathbf{x})\} \rightarrow \{G(\mathbf{y})\}$. In this phase, the end user first generates a key $\mathbf{S}$ which is kept secret at the local side during the whole process. Based on this secret key, the end-user transforms $F(\mathbf{x})$ to its new form $G(\mathbf{y})$ where $\mathbf{y}$ is the new input.
2) **Cloud Computation**: $\mathsf{CloudCom}$ $\{G(\mathbf{y})\} \rightarrow \{\mathbf{y}^*, \Phi\}$. On receiving the transformed problem $G(\mathbf{y})$, the cloud carries out necessary computation and gives the solution $\mathbf{y}^*$ as well as a proof Φ of the validity of the returned solution.
3) **Result Recovery and Verification**: $\mathsf{RecVeri}$ $\{\mathbf{y}^*, \mathbf{S}, \Phi\} \rightarrow \{\mathbf{x}^*, \Lambda\}$. By utilizing the secret key $\mathbf{S}$, the end-user recovers solution $\mathbf{x}^*$ to the original problem from $\mathbf{y}^*$. Based on the proof Φ , the end-user gives the decision $\Lambda = \{\mathsf{Ture}, \mathsf{False}\}$, indicating the validity of $\mathbf{x}^*$.

C. Affine Mapping-Based Problem Transformation

The basic idea of problem transformation is to map the independent variables of the problem to a new group of variables such that the original problem is transformed to a new form. To be specific, suppose the original problem is $F(x)$. We assume that $\psi : \mathbb{R}^n \to \mathbb{R}^n$ is a general one-to-one mapping function. Let $\mathbf{x} = \psi(\mathbf{y})$, then $F(\mathbf{x}) = F(\psi(\mathbf{y})) = (F \circ \psi)(\mathbf{y}) = G(\mathbf{y})$. In this way, the original input $\mathbf{x}$ can be transformed to input $\mathbf{y}$ with the relationship determined by the function ψ. Correspondingly, the original problem $F(x)$ is transformed to a new form $G(\mathbf{y})$ that can be securely outsourced to the cloud. Since the solutions to the two problem satisfy $\mathbf{x}^* = \psi(\mathbf{y}^*)$, the end-user can always recover $\mathbf{x}^*$ from $\mathbf{y}^*$ returned by the cloud. Thus the essence of our proposed scheme lies in finding a proper one-to-one mapping that satisfies the various design goals.

In light of this, we choose ψ as an affine mapping such that $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, where $\mathbf{K} \in \mathbb{R}^{n\times n}$ and $\mathbf{r} \in \mathbb{R}^n$. It is clear that as long as $\mathbf{K}$ is nonsingular, the above mentioned affine mapping is a one-to-one mapping. The correctness of our proposed scheme based on affine mapping is guaranteed by the following theorem.

Theorem 3. *The affine mapping based outsourcing scheme is correct. That is the end-user is guaranteed to recover a valid solution from the solution returned by the cloud.*

For the affine mapping based transformation, the computational complexity mainly lies in the matrix multiplication. In general, the complexity of multiply two $n \times n$ matrices is $\mathcal{O}(n^3)$. Therefore, to limit the computational complexity to $\mathcal{O}(n^2)$, the non-singular matrix $\mathbf{K}$ has to be selected with certain structure while ensuring security of the original problem. In this research, we analyzed four different selections of the non-singular matrix $\mathbf{K}$: (i) $\mathbf{K}$ is a diagonal matrix that has the format $\mathbf{K} = \{k_{ij}|k_{ij} = 0, \forall i \neq j\}$, **(ii)** $\mathbf{K}$ is a permutation matrix that has exactly one non-zero entry in each row and each column in the matrix, (iii) $\mathbf{K}$ is a band matrix that has an upper half-bandwidth p and a lower half-bandwidth q such that $k_{ij} = 0$ for $i > j + p$ and $j > i + q$. The total bandwidth of $\mathbf{K}$ is denoted by $W = p + q + 1$. For simplicity, we assume that $\mathbf{K}$ has an equal upper and lower half-bandwidth $p = q = \omega$, then $W = 2\omega + 1$, and (iv) $\mathbf{K}$ is a sparse matrix with density d which is defined as the ratio of non-zero elements in the matrix. We assume that the number of non-zero elements in each row and each column of $\mathbf{K}$ is up-bounded by a constant θ.

The number of multiplications M in each scheme can be calculated and the results are presented in Table I. In summary, we demonstrate that by selecting special forms of $\mathbf{K}$, the complexity of multiplying $\mathbf{K}$ with an arbitrary matrix $\mathbf{A}$ is $\mathcal{O}(n^2)$. In Table I, we also provided the diffusion and confusion measurement for each scheme, where the diffusion parameter α and confusion parameter β are defined as the number of different entries in $\mathbf{A}$ and the number of different entries in $\mathbf{K}$ that appear in each entry of $\mathbf{A}'$.

Table I
COMPLEXITY AND SECURITY OF EACH SCHEME

Scheme (K)	**Complexity** (M)	**Diffusion** (α)	**Confusion** (β)
Diagonal	n^2	1	1
Permutation	n^2	1	1
Band	Wn^2	W	W
Sparse	θn^2	θ	θ

D. Application to Linear Programming

Consider a linear programming problem denoted by

$$\begin{aligned} &\text{minimize} \quad \mathbf{c}^T\mathbf{x} \\ &\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{D}\mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{16}$$

where $\mathbf{b}, \mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{D} \in \mathbb{R}^{s\times n}$ $(m, s \leq n)$. Under the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, the problem is transformed to

$$\begin{aligned} &\text{minimize} \quad \mathbf{c}^T\mathbf{K}\mathbf{y} + \mathbf{c}^T\mathbf{r} \\ &\text{subject to} \quad \mathbf{AKy{=}b} - \mathbf{Ar}, \mathbf{DKy} \geq -\mathbf{Dr}, \end{aligned}$$

from which we can see that the input $\{\mathbf{c}^T, \mathbf{A}, \mathbf{b}, \mathbf{D}\}$ can be concealed by the secret key $\mathbf{K}$ and $\mathbf{r}$. It is obvious that the computational bottleneck lies in the multiplication of $\mathbf{K}$ with $\mathbf{A}$ and $\mathbf{D}$. Thus the same complexity and security analysis for systems of linear equations applies for linear programming. That is the complexity of the previous four options is all bounded by $\mathcal{O}(n^2)$. In terms of security, the four options are all secure in protecting the input and output while providing different levels of protection of other side information.

E. Application to Non-linear Systems

We consider a system of non-linear equations denoted by $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{F}(\mathbf{x}) = \{f_i(\mathbf{x})|f_i(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \cdots, n\}$. Typically, it is hard to obtain a symbolic solution for the system. Thus the normal method is to solve the system of equations numerically in an iterative way. The main idea is that given a solution $\mathbf{x}_k$ in the k-th iteration, we need to solve the linear system $\partial\mathbf{F}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{F}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$, where $\partial\mathbf{F}(\mathbf{x})$ is the Jacob matrix of $\mathbf{F}(\mathbf{x})$. Then we can obtain the solution $\mathbf{x}_{k+1}$ in the $(k+1)^{th}$ iteration. The iteration will terminate when $|\mathbf{F}(\mathbf{x}^*)| \ll \varepsilon$, where ε is the error tolerance and $\mathbf{x}^*$ is the final solution. To minimize the communication overhead maybe even energy consumption of the end-users, our goal is to design off-line scheme so that the end-users are not required to interact with the cloud except the problem outsourcing and result retrieving process. In this way, the end-users can only focus on a high level view of the problem without knowing the details of problem solving process. Similar to linear programing problem, under the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, the original problem $\mathbf{F}(\mathbf{x})$ can be transformed to $\mathbf{G}(\mathbf{y}) = \{g_i(\mathbf{y})|g_i(\mathbf{y}) = f_i(\mathbf{K}\mathbf{y} + \mathbf{r}) =, i = 1, 2, \cdots, n\}$.

F. Results Verification

For system of linear equations, it is sufficient to verify directly whether $\|\mathbf{A}\mathbf{x}^*\| < \varepsilon$, where ε is a pre-defined error tolerance. The complexity of this verification process is $\mathcal{O}(n^2)$. Under the affine mappings $\mathbf{x} = \mathbf{K}_1\mathbf{y} + \mathbf{r}_1$ and $\mathbf{x} = \mathbf{K}_2\mathbf{z} + \mathbf{r}_2$, the original problem $F(\mathbf{x})$ is transformed to $G(\mathbf{y})$ and $H(\mathbf{z})$, respectively. Then the cloud solves the two outsourced problems and returns the corresponding results $\mathbf{y}^*$ and $\mathbf{z}^*$. The the end-user can utilize the condition $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ as a criterion to verify whether the returned results are valid. The output of a linear programming problem can be divided into three cases: normal, infeasible and unbounded.

The verification scheme works well for the normal case. However, it fails if the cloud simply returns an infeasible result for any outsourced problem. To deal with this issue, we construct a phase I problem [10, Chapter 11] to check the feasibility as follows:

$$\begin{array}{ll} \text{minimize} & \rho \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{D}\mathbf{x} < \rho, \end{array}$$

where ρ is a single variable.

It is obvious that when ρ is large enough, $F_I(\mathbf{x})$ is always feasible. Suppose $\mathbf{x}^*$ minimizes the objective function and ρ^* is the corresponding minimum value. The phase I problem is designed in such a way that when $\rho^* \leq 0$, the original problem $F(\mathbf{x})$ is feasible and $F(\mathbf{x})$ is infeasible otherwise. Thus when the cloud indicates that the solutions to the two outsourced problems $G(\mathbf{y})$ and $H(\mathbf{z})$ are infeasible, it then generates the corresponding phase I problems $G_I(\mathbf{y})$ and $H_I(\mathbf{z})$ and computes the optimal points $\mathbf{y}^*$ and $\mathbf{z}^*$ and the minimum values ρ_G^* and ρ_H^*, respectively. Then at the local side, the verification is the same as that in the normal case.

In the unbounded case, the cloud indicates that the objective function $\mathbf{c}^T\mathbf{x} \to -\infty$ in its domain. To verify the result, we can construct the corresponding *Lagrangian* L as

$$L(\mathbf{x}, \mathbf{u}, \mathbf{v}) = \mathbf{c}^T\mathbf{x} + \mathbf{u}(\mathbf{A}\mathbf{x} - \mathbf{b}) + \mathbf{v}\mathbf{D}\mathbf{x},$$

where $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^s$ are the associated *Lagrange multiplier vectors*. Then based on this Lagrangian $L(\mathbf{x}, \mathbf{u}, \mathbf{v})$, a *Lagrange dual function* can be constructed as

$$\Phi(\mathbf{u}, \mathbf{v}) = \inf_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \mathbf{u}, \mathbf{v}) = \inf_{\mathbf{x} \in \mathcal{D}} \left(\mathbf{c}^T\mathbf{x} + \mathbf{u}(\mathbf{A}\mathbf{x} - \mathbf{b}) + \mathbf{v}\mathbf{D}\mathbf{x}\right),$$

where $\mathcal{D}$ is the domain of the optimization problem. From this definition, it is easy to prove that $\forall \mathbf{v} \succeq 0$, we have the following inequality:

$$\Phi(\mathbf{u}, \mathbf{v}) \leq L(\mathbf{x}^*, \mathbf{u}, \mathbf{v}) \leq \mathbf{c}^T\mathbf{x}^*,$$

where $\mathbf{c}^T\mathbf{x}^*$ denotes the optimal value of the objective function. The above inequality gives a lower bounded of the objective function that depends on the selection of $\mathbf{u}$ and $\mathbf{v}$. Thus, among all the selections of $\mathbf{u}$ and $\mathbf{v}$, find the optimal lower bound is equivalent to solving the following optimization problem:

$$\begin{array}{ll} \text{maximize} & \Phi(\mathbf{u}, \mathbf{v}) \\ \text{subject to} & \mathbf{u} \succeq 0. \end{array}$$

If the original problem is unbounded below, the transformed problem described above should be infeasible since it gives a lower bound of the optimal value in the original problem. Thus the remaining task is to verify the feasibility of the above problem, which has been illustrated in the infeasible case. Let the cloud solve the phase I problems of the two Lagrange dual problems and return the optimal solutions denoted by $(\rho_G^*, \mathbf{y}^*, \mathbf{u}_G^*, \mathbf{v}_G^*)$ and $(\rho_H^*, \mathbf{z}^*, \mathbf{u}_H^*, \mathbf{v}_H^*)$. At the local side, the end-users then check whether $\rho_G^* > 0$ and $\rho_H^* > 0$ and whether the equality $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ holds.

IV. Conclusions

In this paper, we present our research in distributed storage and cloud computing outsourcing, which are two of the most important research topics for cloud computing. In distributed cloud storage, we develop distributed cloud storage that can achieve minimum storage regeneration (MSR), minimum bandwidth regeneration (MBR) and meanwhile, achieving the optimal error correction capability and computational efficiency. For secure computing outsourcing, we develop a cost-aware secure outsourcing (CASO) scheme for general computing problems. We demonstrate that CASO can be widely used to solve linear problems, such as linear programming problems, as well as non-linear problems, in a security and cost-aware trade-off approach. We also developed efficient result verification for various computational problems.

References

[1] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, pp. 4539 – 4551, 2010.

[2] K. Rashmi, N. Shah, K. Ramchandran, and P. Kumar, "Regenerating codes for errors and erasures in distributed storage," in *ISIT 2012*, pp. 1202–1206, 2012.

[3] J. Li, T. Li, and J. Ren, "Beyond the mds bound in distributed cloud storage," in *IEEE INFOCOM 2014*, (Toronto, Canada), April 27 - May 2, 2014.

[4] J. Ren, "On the structure of hermitian codes and decoding for burst errors," *IEEE Transactions on Information Theory*, vol. 50, pp. 2850–2854, 2004.

[5] N. B. Shah, K. V. Rashmi, K. Ramchandran, and P. V. Kumar, "Privacy-preserving and secure distributed storage codes," *http://www.eecs.berkeley.edu/~nihar/publications/privacy_security.pdf/*.

[6] J. Li, T. Li, and J. Ren, "Secure regenerating code," in *IEEE GLOBECOM 2014*, (Austin, TX.), December 8-12, 2014.

[7] K. Rashmi, N. Shah, and P. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," *IEEE Transactions on Information Theory*, vol. 57, pp. 5227–5239, 2011.

[8] J. Li, T. Li, and J. Ren, "Rate-matched regenerating code in hostile networks," in *IEEE ICC 2015*, (London, UK), June 8-12, 2015.

[9] D. Dabiri and I. Blake, "Fast parallel algorithms for decoding reed-solomon codes based on remainder polynomials," *IEEE Transactions on Information Theory*, vol. 41, pp. 873–885, Jul 1995.

[10] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2009.