Heterogeneity Shifts the Storage-Computation Tradeoff in Secure Multi-Cloud Systems

Jiajun Chen, Chi Wan Sung, and Terence H. Chan

***Abstract*—This paper considers the design of heterogeneous multi-cloud systems for big data storage and computing in the presence of cloud collusion and failures. A fundamental concept of such a system is the secrecy capacity, which represents the maximum amount of information that can be stored for each unit of storage space under the requirements of secure distributed computing. A capacity-achieving code is designed for matrix multiplication, a computing subroutine widely used in machine learning applications. The code allows fast parallel decoding and unequal data allocation in the clouds. Such a flexibility leads naturally to the idea of optimizing data allocation to minimize the computing time. Given any feasible storage budget, the optimal solution is derived, characterizing explicitly the fundamental tradeoff between storage and computing. Furthermore, it is shown via majorization theory that the whole tradeoff curve improves if the cloud computing rates are more even. Experiments on Amazon EC2 clusters are conducted, corroborating our theoretical observations and the negligibility of decoding overhead.**

***Index Terms*—Coded distributed computing, heterogeneous systems, multi-cloud computing, secrecy capacity, storage-computation tradeoff.**

I. Introduction

THE availability of massive and inexpensive commodity servers (e.g., Amazon EC2) has made it possible to perform large-scale computing in parallel over a cloud, such as machine learning, graph processing, and big data analytics. Compared with the high cost of establishing and maintaining a datacenter infrastructure, hourly renting these "virtualized" servers to enjoy powerful storage and computing resources is much more effective and economical. There are, however, some major performance bottlenecks in distributed computing scenarios, such as straggler effects [1] and limited bandwidth. Stragglers are distributed nodes which complete their tasks much slower than the average. They lead to serious unpredictable delay due to the individual uncertainty nature of distributed servers. Modern state-of-the-art technologies use coding techniques to add redundant computations to alleviate straggling effects, which have been studied in both homogeneous [2], [3], [4], [5], [6], [7], [8], [9], [10] and heterogeneous [11], [12], [13], [14], [15] computing environments. The idea of such a coding concept is referred to as *minimum latency codes* [16]. By increasing the computation load, another coding concept called *minimum bandwidth codes* [16] is introduced to reduce the communication load, thereby tackling the problem of limited bandwidth. Related works include [2], [17], [18], [19], [20], which utilize codes to deal with the communication bottleneck. Coding techniques have also been used to address security issues in [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], and [32], ensuring no server can steal any information about the confidential data in the information-theoretic sense.

Previous works in distributed computing focus only on a single cloud with a master-worker setup, in which a single master manages a cluster of distributed computational servers. To perform computing tasks, a user needs to send data to the cloud, which incurs substantial delay for large-scale computing. Storing data in the cloud in advance may solve the problem, but it brings risks in terms of security concerns [33]. If one single cloud is used, storing encrypted data involves complicated key management problems. Moreover, *cloud outage*, a critical system failure causing cloud services unavailable for a long time, is a primary concern for cloud computing [34]. It may result from a range of unavoidable causes (e.g., power outage, maintenance and upgrades, etc.), and the sheer scale of a modern cloud datacenter and the pressing internal and external threats make it impossible to completely eliminate the rare occurrence of cloud outages. It should be noted that the nature of cloud outages is very different from stragglers in the classical setting for a cluster of servers: the former are unavailable clouds which cannot provide any service, while the latter are available but slow servers. To address the security and outage issues of a single cloud, it is a plausible solution to employ a multi-cloud strategy.

In this work, we consider joint distributed storage and computation over multiple clouds. Those clouds are heterogeneous with different configurations and computing rates, and some clouds may collude to steal users' data and some may be in outage. A *heterogeneous* (V, K, J) system is constructed, in which distributed storage and computation can be performed reliably and securely over V heterogeneous clouds in the presence of up to $V-K$ cloud outages and J colluding clouds. Our

Manuscript received 8 October 2021; revised 3 June 2022; accepted 3 September 2022. Date of publication 15 September 2022; date of current version 20 January 2023. This work was supported in part by a Grant from the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project CityU 11205318. An earlier version of this paper was presented in part at the 2020 IEEE International Conference on Communications [DOI: 10.1109/ICC40277.2020.9149183]. *(Corresponding author: Chi Wan Sung.)*

Jiajun Chen and Chi Wan Sung are with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong (e-mail: jiajuchen8-c@my.cityu.edu.hk; albert.sung@cityu.edu.hk).

Terence H. Chan is with the Institute for Telecommunications Research, University of South Australia, Adelaide, SA 5095, Australia (e-mail: terence.chan@unisa.edu.au).

Communicated by M. Tao, Associate Editor for Communications.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3206868.

Digital Object Identifier 10.1109/TIT.2022.3206868

model follows the same master-worker architecture, in which a centralized master communicating with multiple distributed workers in a star topology. Our system setup, however, differs from the classical one in the following two major ways:

1) *Deterministic Runtime of a Worker:* For the classical setup, a worker often refers to a computational *server*, whose runtime is random due to stragglers and is commonly assumed to be a shifted-exponential random variable [2] and is generalized to the shifted-Weibull distribution in [12]. In our model, a worker refers to a *cloud*, which consists of multiple servers.When codes are used over these servers, the resultant runtime distribution of a cloud falls within a narrow region with high probability as shown by the empirical measurement on Amazon EC2 clusters [2]. It demonstrates that coded computing is effective in mitigating straggling effects, making the runtime of a single cloud almost deterministic. Therefore, when considering a multi-cloud system with each worker being a cloud, the computing rate of each cloud can reasonably be modeled as a constant parameter related to the cloud configuration (e.g., adopted codes, number of servers, etc.). To the best of our knowledge, there are no results assuming a *general* worker runtime distribution. Existing works all assume a specific runtime distribution (shifted-exponential or shifted-Weibull). Our work assumes deterministic runtime based on measurement results.
2) *Pre-stored Data on Workers:* In the multi-cloud setting, data can be pre-stored into various clouds, which brings new concerns (not considered in the classical setting) on storage cost and data availability. To address these concerns, our setup has additional constraints including total storage budget and reconstruction requirement for data availability. As a result, the computation-time minimization problem in our system is totally different from the traditional formulation. Specifically, we need to consider storage allocation with budget constraint as well as load allocation, since the computing load that can be allocated to a cloud is limited by the amount of data already stored in it. This doubles the dimension of the underlying resource allocation problem from V to $2V$.

Compared with the contributions of previous works on heterogeneous distributed computing, the distinctions of this paper are listed as follows:

- *(Linear Decoding Time)* Our main goal of code design is low decoding complexity. The generator matrix of our proposed code has a special *upper trapezoidal* (UT) structure, which facilitates *linear* decoding time in ordinary cases where all clouds are available. Such an extremely low decoding complexity is achievable because the code is designed in a way which exploits the computing power of the distributed clouds with parallel processing. Moreover, expensive multiplications in decoding are all left at the powerful clouds, while only simple addition and subtraction are required at the user side. This is an especially desirable feature, as clouds have much more powerful computing capability than end users.
- *(Tradeoff between Code Rate and Computation Time)* Existing works on heterogeneous systems [12], [14] focus on *load allocation* for minimizing the expected computation time, without limiting the total amount of load determined by the code rate. In our multi-cloud system, code rate is limited by the storage budget. A lower code rate is more effective in exploiting the computing power of faster clouds but incurs a higher storage cost. Because of this phenomenon, we characterize the fundamental tradeoff between code rate (storage budget) and computation time in heterogeneous systems, rather than just finding the optimal code rate as in [12] and [14]. The result shows that the fundamental tradeoff curve between storage budget and computation time is piecewise linear and convex.
- *(Effects of Heterogeneity)* In heterogeneous multi-cloud systems with different computing service rates, we ask a novel question: how does heterogeneity impact system performance? We adopt the concept of *majorization* [45], which precisely characterizes the degree of "unevenness" of cloud computing rates. Our study reveals that given the total computing rates of all clouds, the more "uneven" the computing rates of individual clouds, the longer the computation time of the system. In other words, heterogeneity negatively impacts the tradeoff between storage and computation: the tradeoff curves of systems with more uneven computing rates (in the sense of majorization) are further away from the origin.

A. Related Works

The problem of storing data securely and reliably in multiple clouds has been investigated in [35], and the proposed storage allocation and coding scheme achieves perfect secrecy with minimum storage cost, but without involving computing issues. Our preliminary study [36] extends the work on distributed storage [35] to incorporate computing. Multi-cloud computing scenarios with high security levels $(J > V - K)$ are considered in [36], in which perfect secrecy is achieved and cloud outages is tackled by using the *nested MDS code*. Storage allocation and computational workload are also optimized to reach the minimum computation time. This work generalizes the results to cases of any security level, and designs a new class of nested MDS codes, called the *upper trapezoidal* (UT) code. Its generator matrix satisfies all the properties of that of a nested MDS code, but has an upper trapezoidal structure for fast decoding. UT code can achieve the secrecy capacity or the minimum computation time in the information-theoretic sense.

There are two works on secure distributed computing that are closely related to our work. In [25], the authors consider secure matrix multiplication across N distributed servers, among which l of them may collude to steal data. Two models are considered: (a) one-sided secure matrix multiplication, in which one matrix is confidential while the other matrix is public to all servers; (b) two-sided secure matrix multiplication, in which both matrices are private. For one-sided secure matrix multiplication, the capacity of secure matrix multiplication, defined as the maximum possible ratio of the desired information and the total information over distributed servers, is proved to be $(N-l)/N$. Our work extends this result to cases with $V-K$ unavailable distributed nodes. In our (V,K,J) system, the secrecy capacity, which represents the maximum amount of information that can be stored for each unit of storage space for secure distributed computing, in the presence of up to $V-K$ cloud outages and J colluding clouds, is shown to be $(K-J)/V$. The result of one-sided secure matrix multiplication in [25] corresponds to our special case with $V=K$. Another related work [24] proposes Lagrange Coded Computing (LCC), a distributed computing framework on a cluster of distributed servers, which can simultaneously provide resiliency, security, and privacy. LCC generalizes prior works from linear computations to arbitrary multivariate polynomial computations. One of the main differences of our model from the one in [24] is that we consider a heterogeneous framework composed of multiple clouds, in which the encoded data can be unequally allocated to the clouds. Furthermore, we characterize the tradeoff between storage and computation time in the information-theoretical sense.

In some sense, our (V,K,J) system can be regarded as a generalization of the *erasure-erasure wiretap channel-II* [37]. In the erasure-erasure wiretap channel of parameters (n,m,v), the transmitter sends n symbols to the legitimate receiver who can only receive m symbols after experiencing random erasures. The eavesdropper can observe arbitrary v symbols out of the transmitted n symbols. The secrecy capacity in the wiretap channel, which is defined as the maximum amount of secret information that can be conveyed, is proved to be $m-v$ assuming $m \geq v$ [37], [38]. If we normalize the wiretap channel secrecy capacity by n, then the secrecy capacities of the two systems are the same. Adding the computing requirement in our multi-cloud system does not change the secrecy capacity because the computing task on the data stored in our system is assumed to be public. Allowing unequal storage allocation in different clouds does not change it either because the secrecy capacity is shown to be reached when the allocation is equal.

Decoding complexity for coded distributed computing is an important issue because its overhead may outweigh the benefit of coded computing. The idea of exploiting codes to speed up large-scale computing was initially proposed in [2], which apply an (n,k) MDS code to mitigate $n-k$ stragglers in a homogeneous cloud, resulting short *computation time*, but requiring long *decoding time* when k becomes large. To reduce the decoding time, some existing works divide workers into multiple groups and then employ an individual MDS code for each group [4], [15], while another work [39] removes expensive multiplications and divisions from both the encoding and decoding phases by combining shift-and-addition and zigzag decoding. In [5] and [12], peeling decoder is used, which reduces the decoding complexity to nearly linear at the cost of slightly higher recovery threshold. Taking data security into consideration, the decoding process becomes more elaborate. The authors in [27] use cross subspace alignment (CSA) based coding scheme to achieve the secrecy capacity for distributed batch matrix multiplication, which requires a cubic decoding complexity if the inverse of the decoding matrix is obtained simply by Gaussian elimination. In [24], Lagrange Coded Computing (LCC) is proposed in a single cloud, which requires a decoding complexity of $O(R\log^2 R\log\log R)$ for linear computations, where R is the minimum number of servers required to restore the computational result (i.e., recovery threshold). If LCC is applied to our (V,K,J) system, only the computation results from K clouds are used for decoding even if there are more than K available clouds, resulting in the decoding complexity of $O(K\log^2 K\log\log K)$. Similar codes, constructed using polynomials for secure matrix multiplication in [25] and [26], have the same decoding complexity of $O(\eta\log^2\eta\log\log\eta)$, where η is the recovery threshold. In multi-cloud scenarios, the cases where all clouds are available are ordinary, since cloud outages rarely happen. Our proposed UT code can achieve an overall decoding complexity of $O(JK)$, which can be implemented in linear time, $O(K)$, by allowing expensive multiplications to be calculated in parallel by the clouds. Users just need to do addition and subtraction to decode the computing result. This is a particularly desirable feature, as clouds are considered to have much more powerful computing capability than users. Therefore, UT code is more suitable for multi-cloud linear computations in terms of decoding time.

B. Organization

The remainder of this paper is organized as follows. Section II presents the system model for distributed storage and computation. Section III shows our main results. In Section IV, two necessary conditions for storage allocation are derived. Our designed coding scheme is proved to achieve the secrecy capacity with linear decoding time in Section V. In Section VI, an optimization problem of computation time under a given storage budget is formulated and solved, and the majorization property is described. Numerical studies and experiment results are provided in Section VII and Section VIII offers some concluding remarks.

Notation: Define $\mathcal{V} \triangleq \{1,2,\dots,V\}$ for $V \in \mathbb{N}$. Let $\mathcal{J} \subset_J \mathcal{V}$ represent that $\mathcal{J}$ is a subset of $\mathcal{V}$ with cardinality J, $\mathbf{0}_n$ be the all-zero vector of length n, and $\mathbf{1}_n$ be the all-one vector of length n. For any real number x, define $(x)^+ \triangleq \max(x,0)$. For two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, define $\boldsymbol{x} \leq \boldsymbol{y}$ if $x_i \leq y_i$ for all i, and $\boldsymbol{x} < \boldsymbol{y}$ if $x_i < y_i$ for all i.

II. System Model

Consider a setting where a user wants to store an $m \times s$ confidential matrix $\mathbf{A}$ in a (V,K,J) system and perform distributed matrix-vector computations securely. Each of its rows can be regarded as a data record while the width s can be regarded as the number of features associated with each data record. The entries of $\mathbf{A}$ are independent and identically distributed (i.i.d.) random variables, each of which is drawn uniformly from a sufficiently large finite field[1] of size q, denoted by $\mathbb{F}_q$. The entropy of $\mathbf{A}$, $H(\mathbf{A})$, is equal to $ms\log_2 q$ bits.

A. Storage Model

In the (V,K,J) system, it is required that distributed storage and computing can be done over V clouds in the presence of

[1]This assumption of uniform i.i.d. entries of $\mathbf{A}$ is needed for proving the converse, but not for the achievability of our proposed code.

up to $V-K$ cloud outages and J colluding clouds. To ensure security and reliability, the confidential matrix $\mathbf{A} \in \mathbb{F}_q^{m\times s}$ is encoded into $\tilde{\mathbf{A}} \in \mathbb{F}_q^{n\times s}$ and then allocated to V clouds for distributed storage and computing. The code is said to be an (n,m) code and its rate R is given by m/n, where $n \geq m$. Let S be the storage budget for each row of $\mathbf{A}$, i.e., $n \leq mS$. In other words, the system is required to operate with code rate above $1/S$.

The encoded matrix $\tilde{\mathbf{A}}$, of n rows, can be partitioned into V submatrices as follows:

$$\tilde{\mathbf{A}} \triangleq \begin{bmatrix} \tilde{\mathbf{A}}_1 \\ \tilde{\mathbf{A}}_2 \\ \vdots \\ \tilde{\mathbf{A}}_V \end{bmatrix}, \tag{1}$$

where $\tilde{\mathbf{A}}_i \in \mathbb{F}_q^{l_i\times s}$ and $n=\sum_{i=1}^{V} l_i$. There is a cloud manager, which distributes $\tilde{\mathbf{A}}_i$ to Cloud i, for $i=1,2,\ldots,V$. We call $\boldsymbol{l} \triangleq (l_1,l_2,\ldots,l_V)$ the *storage vector*, where l_i represents the number of encoded rows allocated to Cloud i.

The criterion for data security is the same as that in the secret sharing problem. We call it *secrecy requirement I*, which ensures no information would be disclosed to any J colluding clouds $(J<V)$ in the information-theoretic sense. For a set $\mathcal{S}$, define $\tilde{\mathbf{A}}_{\mathcal{S}}$ as the submatrix of $\tilde{\mathbf{A}}$ obtained by preserving $\tilde{\mathbf{A}}_i$ for all $i \in \mathcal{S}$. Then the perfect secrecy requirement can be written as

$$I(\mathbf{A};\tilde{\mathbf{A}}_{\mathcal{J}})=0,\ \forall \mathcal{J} \subset_J \mathcal{V}. \tag{2}$$

To ensure data reliability, it is also required that the cloud manager can retrieve the original matrix $\mathbf{A}$ from any K available clouds $(J<K\leq V)$. This *reconstruction requirement* for distributed storage can be expressed as

$$H(\mathbf{A}|\tilde{\mathbf{A}}_{\mathcal{K}})=0,\ \forall \mathcal{K} \subset_K \mathcal{V}. \tag{3}$$

We refer it as a (V,K,J) system, where K and J represent the desired reliability and security levels, respectively. Fig. 1 shows an example of a $(5,4,3)$ system.

B. Computation Model

In our model, each cloud consists of multiple servers. By applying coded computing, its runtime is of little randomness [2]. Therefore, in our V-cloud system, the service rate of Cloud i, $i\in\mathcal{V}$, is modeled as a constant parameter μ_i only related to the cloud configuration. Denote the computing rates of V clouds by a vector $\boldsymbol{\mu} \triangleq (\mu_1,\ldots,\mu_V)\in\mathbb{R}^V$. To study the effect of heterogeneous computing rates on computation time, we use the notion of *majorization* [45] to characterize the uneveness of computing rates of the clouds.

Definition 1 (Majorization [45]): : Let $\boldsymbol{\mu}^{(1)},\boldsymbol{\mu}^{(2)}\in\mathbb{R}^V$ be two vectors whose components are sorted in descending order. We say that $\boldsymbol{\mu}^{(1)}$ majorizes $\boldsymbol{\mu}^{(2)}$, written as $\boldsymbol{\mu}^{(1)}\succ\boldsymbol{\mu}^{(2)}$, if

$$\sum_{i=1}^{V}\mu_i^{(1)}=\sum_{i=1}^{V}\mu_i^{(2)},\ \text{and}\ \sum_{i=p}^{V}\mu_i^{(1)}\leq\sum_{i=p}^{V}\mu_i^{(2)}$$

for $p=2,3,\ldots,V$.

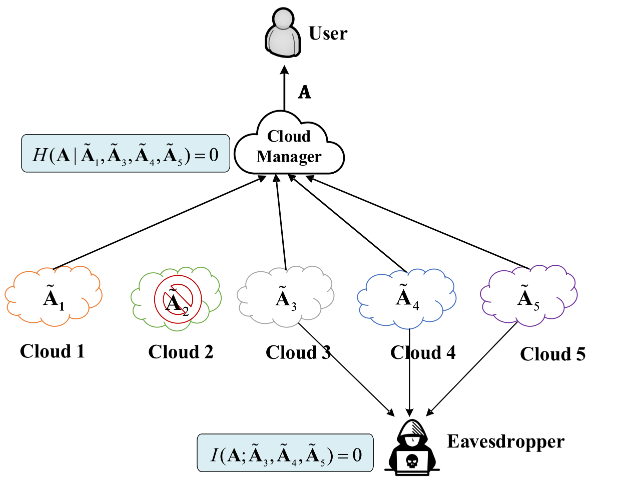

Fig. 1. In this $(5,4,3)$ system, a cloud outage occurs in Cloud 2 and its service is unavailable. However, the user can still retrieve the original data from the other four clouds. Clouds 3, 4 and 5 collude to steal the confidential data, but no information is leaked to the eavesdropper.

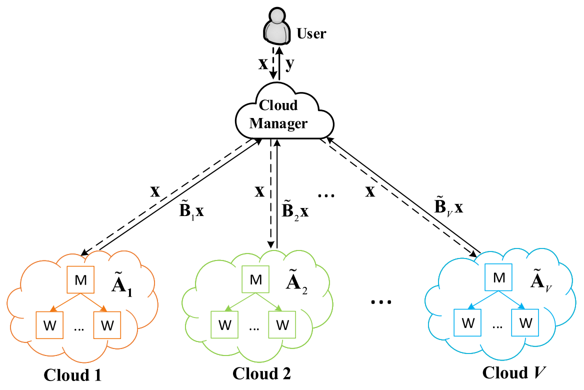

Fig. 2. Cloud i stores $\tilde{\mathbf{A}}_i$ with l_i rows. The master (M) calculates the product of the $\tilde{\mathbf{B}}_i$, consisting of r_i $(0\leq r_i\leq l_i)$ rows from $\tilde{\mathbf{A}}_i$, and the input vector $\mathbf{x}$ over distributed workers (W). Once the cloud manager receives enough calculated rows, it can restore the original product $\mathbf{y}$.

Consider the common computation task of matrix-vector multiplication $\mathbf{y}=\mathbf{A}\mathbf{x}$, where the input vector $\mathbf{x}\in\mathbb{F}_q^{s\times 1}\backslash\{\mathbf{0}\}$ is deterministic and publicly known, thus not requiring to be secure. Since the entries of $\mathbf{A}$ are independently and uniformly distributed in $\mathbb{F}_q$, so are the entries of $\mathbf{y}$ for any non-zero $\mathbf{x}$. Different from distributed storage, the computing task does not have to be completed only in K clouds. Instead, we can utilize all available clouds to compute, speeding up the computation. While Cloud i stores $\tilde{\mathbf{A}}_i$ with l_i rows, in general, it may only use r_i rows $(0\leq r_i\leq l_i)$ for computing, and its runtime is r_i/μ_i. If Cloud i is in outage, then $r_i=0$. Denote the *load vector* by $\boldsymbol{r}\triangleq(r_1,r_2,\ldots,r_V)$. Let $\tilde{\mathbf{B}}_i$ be a submatrix of $\tilde{\mathbf{A}}_i$ with r_i rows. After computation, $\tilde{\mathbf{B}}_i\mathbf{x}$ is sent to the cloud manager. The computation model is illustrated in Fig. 2. After collecting the computing results $\tilde{\mathbf{B}}_1\mathbf{x}$, $\tilde{\mathbf{B}}_2\mathbf{x}$, $\ldots$, $\tilde{\mathbf{B}}_V\mathbf{x}$, the cloud manager stacks them up to form $\tilde{\mathbf{y}}=\tilde{\mathbf{B}}_{\mathcal{V}}\mathbf{x}$, where $\tilde{\mathbf{B}}_{\mathcal{V}}$ is the matrix obtained by collecting $\tilde{\mathbf{B}}_i$ for all $i\in\mathcal{V}$. The vector $\tilde{\mathbf{y}}$ is a subvector of $\tilde{\mathbf{A}}\mathbf{x}$, and its length is $\|\boldsymbol{r}\|_1=\sum_{i=1}^{V} r_i$, where $\|\cdot\|_1$ is the ℓ_1 norm.

The desired $\mathbf{y}$ needs to be decodable from $\tilde{\mathbf{y}}$ in the presence of at most $V-K$ cloud outages. Since which clouds will be in outage is not known in advance, we need to ensure that the result can be computed from the stored data of any K clouds.

This *computing requirement* is expressed as

$$H(\mathbf{y}|\tilde{\mathbf{y}}) = 0,\ \forall \mathbf{x} \in \mathbb{F}_q^{s\times 1} \setminus \{\mathbf{0}\},\ \forall \boldsymbol{r} \text{ s.t. } \|\boldsymbol{r}\|_1 = \min_{\mathcal{K}\subset_K \mathcal{V}} \sum_{i\in\mathcal{K}} l_i, \tag{4}$$

where the constraint on $\|\boldsymbol{r}\|_1$ ensures that the computing task can be completed in the presence of at most $V-K$ cloud outages. In addition, the computing result is required to be perfectly secure by avoiding J colluding clouds, i.e.,

$$I(\mathbf{Ax}; \tilde{\mathbf{B}}_{\mathcal{J}}\mathbf{x}) = 0,\ \forall \mathcal{J} \subset_J \mathcal{V}, \forall \mathbf{x} \in \mathbb{F}_q^{s\times 1} \setminus \{\mathbf{0}\}. \tag{5}$$

We call it *secrecy requirement II*. It should be noted that secrecy requirement II is implied by secrecy requirement I, and reconstruction requirement by computing requirement. (The proof is in Lemma 14.) For this reason, we may just consider two stronger system requirements (2) and (4) in the following definitions.

Definition 2 (Achievable Rate): A rate R is said to be achievable if there exists, for sufficiently large n, an (n, Rn) code such that the system requirements in (2) and (4) are both satisfied.

Definition 3 (Secrecy Capacity): The secrecy capacity of the (V, K, J) system, denoted by C, is defined as the supremum of all achievable rates.

The secrecy capacity specifies how much information can be stored in the (V, K, J) system for secure distributed computing such that all the system requirements are fulfilled. However, it does not concern the computation time, which, in practice, is an important performance measure. We define computation time as the runtime for computing:

Definition 4 (Computation Time): Given a load vector $\boldsymbol{r}$ in the (V, K, J) system, the (normalized) computation time is defined as $T \triangleq \max_{i\in\mathcal{V}} \frac{r_i}{m\mu_i}$.

Definition 5: A storage-computation pair $(S, T) \in \mathbb{R}^2$ is said to be *feasible* if for any $\zeta > 0$ and sufficiently large n, there exists an (n, Rn) code with code rate $R \geq 1/S$, allocation vector $\boldsymbol{l}$ with $\|\boldsymbol{l}\|_1 = n$, and load vector $\boldsymbol{r} \leq \boldsymbol{l}$ that achieve a storage-computation pair (S', T') such that $|S - S'| \leq \zeta$, $|T - T'| \leq \zeta$, and the system requirements in (2) and (4) are both satisfied.

Definition 6 (Optimal Tradeoff): Given a storage budget S, the minimum computation time $T^*(S)$ of a (V, K, J) system is defined as

$$T^*(S) \triangleq \inf\{T : (S, T) \text{ is feasible}\}. \tag{6}$$

$T^*(S)$ characterizes the optimal tradeoff between storage budget and computation time in a multi-cloud scenario. Since the storage budget S constrains the code rate to be above the threshold of $1/S$, for the range of S such that the threshold is smaller than the secrecy capacity, the tradeoff is well defined and answers the question of how much the computation time can be reduced at the cost of a lower code rate than secrecy capacity.

Remark 1: It should be noted that (S, T) may be feasible at load vector $\boldsymbol{r}$ that has zero components, meaning the corresponding clouds are not used for computing. On the other hand, to achieve the minimum computation time $T^*(S)$, all the components of the load vector $\boldsymbol{r}$ must be non-zero, meaning all the clouds are exploited for computation. We will explain that in detail in Section VI.

III. Main Results

This section presents three fundamental results obtained in this study. First, the secrecy capacity shown in Theorem 1 can be achieved by the upper trapezoidal (UT) code, which can be implemented in linear time by parallel decoding. Second, the optimal tradeoff between storage and computation is characterized in Theorem 2. Finally, the impact of heterogeneity on the tradeoff curve is stated in Theorem 3.

Theorem 1: The proposed UT code achieves the secrecy capacity of the (V, K, J) system,

$$C = \frac{K-J}{V}, \tag{7}$$

with a decoding complexity of $O(JK)$ in the ordinary no-outage case. With parallel decoding by multiple clouds, the decoding complexity can be reduced to $O(K)$.

The proof of Theorem 1 consists of two parts, an upper bound on secrecy capacity in Corollary 6 in Section IV and the achievability of that bound with its decoding complexity analysis in Section V. Although the capacity can be easily achieved by many simple codes, like the nested MDS code and random codes, they require long decoding time as the system becomes large. To reduce the decoding time, UT code is designed with a decoding complexity of $O(JK)$ in the ordinary case where cloud outages do not occur. It is called "UT" code because its generator matrix has a permuted *upper trapezoidal* structure, which allows the expensive multiplications of the decoding process to be computed in parallel by the clouds in linear time, $O(K)$. For end users, they only need to do addition and subtraction to decode the computing result. This is an especially desirable feature, as clouds have much more powerful computing capability than end users. TABLE I compares the decoding complexity of proposed coding scheme to prior works on secure matrix multiplication. Our code outperforms existing codes due to the novel idea of using distributed clouds to decode in parallel. Notice that decoding is itself a computing task. It is appealing to exploit the computing power of the distributed clouds to perform decoding as well as matrix multiplication.

To achieve the secrecy capacity in (7), the encoded data should be *equally* allocated to multiple clouds. This is the most space-efficient way to satisfy all system requirements. In *heterogeneous* multi-cloud systems, however, equal storage allocation is not optimal because when the fastest cloud finishes computing with all its local data, other slower clouds still need to compute as their tasks are not finished yet. If we can transfer some data from the slower clouds to the fastest cloud without violating the security requirement, the overall computation time can be reduced. But such an unequal storage allocation inevitably reduces the code rate. In other words, it is possible to reduce the computation time at the expense of more storage space. The exact relationship between computation time, T, and storage budget, S, is shown below:

TABLE I

COMPARISONS OF DECODING COMPLEXITY IN A DISTRIBUTED COMPUTING SYSTEM WITH V NODES, WHERE THE RECOVER THRESHOLD REPRESENTS THE MINIMUM NUMBER OF NODES REQUIRED TO RESTORE THE COMPUTATIONAL RESULT

Coding Scheme	Recover Threshold	Decoding Complexity
Codes constructed using polynomials [24]–[26]	$\eta < V$	$O(\eta \log^2 \eta \log\log \eta)$
CSA based scheme[2] [27]	V	$O(V^3)$
UT code	K	$O(K)$ with parallel decoding

Theorem 2: The fundamental tradeoff between storage budget and computation time, i.e., $T^*(S)$, is a piecewise-linear, convex, decreasing function of S and becomes flat when S exceeds a threshold. The explicit form for $T^*(S)$ is shown in Proposition 18.

To prove Theorem 2, we first derive a lower bound on $T^*(S)$ in the information-theoretic sense in Section VI, and then show that the UT code can achieve that bound when the load vector has no zero elements, i.e., when there is no cloud outage and the system computing resource can be totally exploited. The explicit form for that lower bound is derived by solving an optimization problem, as shown in Section VI-C. Our result reveals that as the storage budget increases, the graph for the optimal computation time decreases in a piecewise-linear manner and eventually becomes flat. The decreasing rate also drops with the growth of storage budget, so $T^*(S)$ is convex. Note that there is a reciprocal relationship between storage budget and code rate: (a) the storage budget constrains the code rate to be above a certain level; (b) the minimum storage budget that makes the optimization problem feasible corresponds to the maximum achievable code rate (i.e., secrecy capacity). Hence, the derived tradeoff also reveals the relationship between code rate and computation time. In other words, given a feasible storage budget, the minimum computation time is usually reached with unequal allocations by UT code, at the cost of a lower code rate than secrecy capacity.

Theorem 3: For any two systems with computing rate vectors $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$ such that $\boldsymbol{\mu}^{(1)} \succ \boldsymbol{\mu}^{(2)}$, we must have $T_1^* \geq T_2^*$ under the same storage budget, where T_1^* and T_2^* are the optimal computation times of the two systems, respectively.

Theorem 3 is a direct consequence of Proposition 20 in Section VI. This result inspires us on what an efficient multi-cloud system looks like. If the total computing power is fixed, homogeneous multi-cloud systems provide faster computing service than heterogeneous systems. In general, the more even the computing rates are, the shorter the computing time a system can achieve. The reason is that it is harder to fully utilize the fast clouds if the computing rates are more uneven.

IV. NECESSARY CONDITIONS FOR STORAGE ALLOCATION

In this section, we first show that secrecy requirement II is a weaker condition than secrecy requirement I and computing requirement implies reconstruction requirement. Then a necessary condition that satisfy the two stronger requirements can be obtained, which is useful to derive the lower bound on the minimum computation time in Section VI. Another weaker necessary condition meeting security requirement I and reconstruction requirement gives an upper bound on the secrecy capacity, which has been demonstrated in previous works [25], [26], [27].

In the (V, K, J) system, the confidential matrix is encoded with code rate m/n, where m is the number of rows of the original matrix and n is the number of rows of the encoded matrix. The encoded matrix is partitioned and assigned to V clouds according to storage allocation vector $\boldsymbol{l}$ (with $\|\boldsymbol{l}\|_1 = n$) such that all the system requirements are fulfilled. The following lemma shows the relationships between various system requirements:

Lemma 4: (i) Secrecy requirement II is implied by secrecy requirement I.

(ii) Reconstruction requirement is implied by computing requirement.

Proof: (i) We find that $\mathbf{Ax} \to \mathbf{A} \to \tilde{\mathbf{A}}_{\mathcal{J}}$ forms a Markov chain, because

$$I(\mathbf{Ax}; \tilde{\mathbf{A}}_{\mathcal{J}} | \mathbf{A}) = H(\tilde{\mathbf{A}}_{\mathcal{J}} | \mathbf{A}) - H(\tilde{\mathbf{A}}_{\mathcal{J}} | \mathbf{A}, \mathbf{Ax}) \overset{(a)}{=} 0,$$

where (a) holds because $\mathbf{Ax}$ is a function of $\mathbf{A}$. Similarly, it can be shown that $(\mathbf{Ax}, \mathbf{A}) \to \tilde{\mathbf{A}}_{\mathcal{J}} \to \tilde{\mathbf{B}}_{\mathcal{J}}\mathbf{x}$ also forms a Markov chain, since $\tilde{\mathbf{B}}_{\mathcal{J}}\mathbf{x}$ is a function of $\tilde{\mathbf{A}}_{\mathcal{J}}$. Hence, $\mathbf{Ax} \to \mathbf{A} \to \tilde{\mathbf{A}}_{\mathcal{J}} \to \tilde{\mathbf{B}}_{\mathcal{J}}\mathbf{x}$ forms a Markov chain.

Then, we have

$$0 \leq I(\mathbf{Ax}; \tilde{\mathbf{B}}_{\mathcal{J}}\mathbf{x}) \overset{(b)}{\leq} I(\mathbf{A}; \tilde{\mathbf{A}}_{\mathcal{J}}) \overset{(c)}{=} 0,$$

where (b) follows from the Data Processing Inequality [41, p.31], and (c) follows from the secrecy requirement I. Therefore, secrecy requirement I implies secrecy requirement II.

(ii) By the computing requirement, we have

$$0 = H(\mathbf{Ax} | \tilde{\mathbf{B}}_{\mathcal{V}}\mathbf{x}) \geq H(\mathbf{Ax} | \tilde{\mathbf{B}}_{\mathcal{V}}\mathbf{x}, \tilde{\mathbf{B}}_{\mathcal{V}}) \overset{(d)}{=} H(\mathbf{Ax} | \tilde{\mathbf{B}}_{\mathcal{V}}) \geq 0,$$

where (d) comes form $\tilde{\mathbf{B}}_{\mathcal{V}}\mathbf{x}$ being a function of $\tilde{\mathbf{B}}_{\mathcal{V}}$. Therefore, we have $H(\mathbf{Ax} | \tilde{\mathbf{B}}_{\mathcal{V}}) = 0$ for any load vector $\|\boldsymbol{r}\| = \min_{\mathcal{K} \subset_K \mathcal{V}} \sum_{i \in \mathcal{K}} l_i$, implying $H(\mathbf{Ax} | \tilde{\mathbf{A}}_{\mathcal{K}}) = 0$, $\forall \mathbf{x} \in \mathbb{F}_q^{s \times 1} \setminus \{\mathbf{0}\}$.

If $\mathbf{x} = [1, 0, \ldots, 0]^T$, then $\mathbf{Ax}$ is the first column of $\mathbf{A}$, denoted by $\boldsymbol{a}_1$. Thus, we have $H(\boldsymbol{a}_1 | \tilde{\mathbf{A}}_{\mathcal{K}}) = 0$. Similarly, if $\mathbf{x}$ is the i-th standard basis of the s-dimensional vector space, $\mathbf{Ax}$ becomes the i-th column of $\mathbf{A}$ (i.e., $\boldsymbol{a}_i$), resulting in $H(\boldsymbol{a}_i | \tilde{\mathbf{A}}_{\mathcal{K}}) = 0$. Therefore, $H(\mathbf{A} | \tilde{\mathbf{A}}_{\mathcal{K}}) = H(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_s | \tilde{\mathbf{A}}_{\mathcal{K}}) = 0$. □

To meet the two stronger requirements (i.e., secrecy requirement I and computing requirement), a necessary condition is obtained in Theorem 5, which can be used to obtain the lower bound on the computation time.

[2]The authors of [27] prove the decodability of the CSA coding scheme by showing the decoding matrix invertible. Here we assume the inverse is obtained by Gaussian elimination.

Theorem 5 (Necessary Condition): The system requirements (2), (3), (4) and (5) can be met only if the storage allocation vector $\boldsymbol{l}$ satisfies

$$\min_{\mathcal{K}\subset_K\mathcal{V}}\sum_{i\in\mathcal{K}} l_i - \max_{\mathcal{J}\subset_J\mathcal{V}}\sum_{i\in\mathcal{J}} l_i \geq m > 0. \tag{8}$$

Proof: According to Lemma 14, we can ignore security requirement II and reconstruction requirement, since they cannot tighten the necessary condition in (8).

By the computing requirement, for any arbitrary $\mathcal{J}\subset_J\mathcal{V}$, we have

$$\begin{aligned} H(\mathbf{y}) &= I(\mathbf{y};\tilde{\mathbf{y}}) \\ &= I(\mathbf{A}\mathbf{x};\tilde{\mathbf{B}}_{\mathcal{V}}\mathbf{x}) \\ &= I(\mathbf{A}\mathbf{x};\tilde{\mathbf{B}}_{\mathcal{J}}\mathbf{x},\tilde{\mathbf{B}}_{\mathcal{V}\setminus\mathcal{J}}\mathbf{x}) \\ &= I(\mathbf{A}\mathbf{x};\tilde{\mathbf{B}}_{\mathcal{J}}\mathbf{x}) + I(\mathbf{A}\mathbf{x};\tilde{\mathbf{B}}_{\mathcal{V}\setminus\mathcal{J}}\mathbf{x}|\tilde{\mathbf{B}}_{\mathcal{J}}\mathbf{x}) \\ &\overset{(a)}{=} I(\mathbf{A}\mathbf{x};\tilde{\mathbf{B}}_{\mathcal{V}\setminus\mathcal{J}}\mathbf{x}|\tilde{\mathbf{B}}_{\mathcal{J}}\mathbf{x}) \\ &\leq H(\tilde{\mathbf{B}}_{\mathcal{V}\setminus\mathcal{J}}\mathbf{x}), \end{aligned} \tag{9}$$

where (a) comes from Lemma 14. Since $H(\mathbf{y}) = m\log_2 q$, (9) implies that

$$m\log_2 q \leq H(\tilde{\mathbf{B}}_{\mathcal{V}\setminus\mathcal{J}}\mathbf{x}) \leq \sum_{i\in\mathcal{V}\setminus\mathcal{J}} H(\tilde{\mathbf{B}}_i\mathbf{x}) \leq \sum_{i\in\mathcal{V}\setminus\mathcal{J}} r_i\log_2 q, \tag{10}$$

which holds for any load vector $\boldsymbol{r}$ such that $\|\boldsymbol{r}\|_1 = \min\limits_{\mathcal{K}\subset_K\mathcal{V}}\sum_{i\in\mathcal{K}} l_i$. In other words, for any load vector $\boldsymbol{r}$ with required length, we have

$$m \leq \sum_{i\in\mathcal{V}} r_i - \max_{\mathcal{J}\subset_J\mathcal{V}}\sum_{i\in\mathcal{J}} r_i = \min_{\mathcal{K}\subset_K\mathcal{V}}\sum_{i\in\mathcal{K}} l_i - \max_{\mathcal{J}\subset_J\mathcal{V}}\sum_{i\in\mathcal{J}} r_i.$$

If

$$\min_{\mathcal{K}\subset_K\mathcal{V}}\sum_{i\in\mathcal{K}} l_i \leq \max_{\mathcal{J}\subset_J\mathcal{V}}\sum_{i\in\mathcal{J}} l_i, \tag{11}$$

then the minimum upper bound becomes zero, since $\boldsymbol{r}$ can be chosen such that $r_i = 0$ for all $i\in\mathcal{V}\setminus\mathcal{J}$. Otherwise, the minimum upper bound is obtained when $\boldsymbol{r}$ is chosen such that $r_i = l_i$ for all $i\in\mathcal{J}$. Combining the two cases, we have

$$m \leq \big(\min_{\mathcal{K}\subset_K\mathcal{V}}\sum_{i\in\mathcal{K}} l_i - \max_{\mathcal{J}\subset_J\mathcal{V}}\sum_{i\in\mathcal{J}} l_i\big)^+, \tag{12}$$

which completes the proof. □

Corollary 6 (Converse of Theorem 1): The secrecy capacity of the (V,K,J) system is bounded by

$$C \leq \frac{K-J}{V}, \tag{13}$$

where the equality holds only if the storage allocation is equal.

Proof: By Theorem 5, we have the following weaker necessary condition:

$$m \leq \min_{\mathcal{J}\subset_J\mathcal{K}\subset_K\mathcal{V}}\Big(\sum_{i\in\mathcal{K}} l_i - \sum_{i\in\mathcal{J}} l_i\Big), \tag{14}$$

which is equivalent to

$$m \leq \sum_{i\in\mathcal{I}} l_i, \quad \forall\mathcal{I}\subset_{K-J}\mathcal{V}. \tag{15}$$

Since there are $\binom{V}{K-J}$ possible subsets $\mathcal{I}$ of size $V-J$, we sum up the inequalities and have

$$\binom{V-1}{K-J-1}\sum_{i\in\mathcal{V}} l_i \geq \binom{V}{K-J}m,$$

i.e., $m/n \leq (K-J)/V$. Hence, the supremum of all achievable rates, i.e., secrecy capacity C, is bounded above by $(K-J)/V$. To achieve the capacity bound, (15) must hold with equality for all $\mathcal{I}\subset_{K-J}\mathcal{V}$, i.e., the storage allocation must be equal for each cloud. □

Remark 2: The capacity bound in (13) is implied by the weaker necessary condition in (14), which has been proved to follow from security requirement I and reconstruction requirement in [35]. It should be noted that Corollary 6 has been observed by previous works [25], [26], [27]. The necessary condition to achieve that bound is all l_i's being equal, in which case the problem is similar to secret sharing.

V. Code Construction

In the (V,K,J) system, the user is accessible to at least K submatrices of $\tilde{\mathbf{A}}$ from V clouds while the eavesdropper to at most J submatrices. In this section, we design a code called upper trapezoidal (UT) code, which encodes a confidential matrix $\mathbf{A}\in\mathbb{F}_q^{m\times s}$ into $\tilde{\mathbf{A}}_i\in\mathbb{F}_q^{l_i\times s}$ for Cloud i, where $i=1,2,\ldots,V$ and $\|\boldsymbol{l}\|_1 = n$. For sufficiently large n, UT code can satisfy all the system requirements stated in Section II and achieve the secrecy capacity.

A. Coding Scheme: UT Code

Our proposed coding scheme has a block-by-block structure. By dividing $\mathbf{A}$ into $\hat{m}$ submatrices of equal size[3],i.e.,

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_1\\ \vdots \\ \mathbf{A}_{\hat{m}}\end{bmatrix},$$

we call each submatrix $\mathbf{A}_i$ with b rows a *block*, where $b\triangleq\frac{m}{\hat{m}}$ is the block size. The $\hat{m}$ input blocks are encoded into $\hat{n}\triangleq n/b$ blocks according to a preset parameter pair $(\hat{\boldsymbol{l}},\hat{\boldsymbol{r}})\in\mathbb{N}^V\times\mathbb{N}^V$, where $\|\hat{\boldsymbol{l}}\|_1 = \hat{n}$, $\hat{\boldsymbol{r}}\leq\hat{\boldsymbol{l}}$ and $\|\hat{\boldsymbol{r}}\|_1 = \min\limits_{\mathcal{K}\subset_K\mathcal{V}}\sum_{i\in\mathcal{K}}\hat{l}_i$. The vector $\hat{\boldsymbol{l}}$ is the encoding parameter related to storage allocation, i.e., the encoded data are allocated to V clouds according to the allocation vector $\boldsymbol{l} = b\hat{\boldsymbol{l}}$. The vector $\hat{\boldsymbol{r}}$ is the encoding parameter related to computing load. Specifically, if there are no cloud outages, let the computing load vector be $\boldsymbol{r} = b\hat{\boldsymbol{r}}$. Otherwise, clouds in outage cannot compute and their computing load becomes zero. A feasible computing load vector $\boldsymbol{r}$ such that $\boldsymbol{r}\leq\boldsymbol{l}$, $\|\boldsymbol{r}\|_1 = \min\limits_{\mathcal{K}\subset_K\mathcal{V}}\sum_{i\in\mathcal{K}} l_i$, and whose support does not involve any cloud in outage will be used. Cloud i, storing l_i/b blocks, then uses r_i/b blocks for computing.

We design an $(\hat{n},\omega)$ linear code with parameters $(\hat{\boldsymbol{l}},\hat{\boldsymbol{r}},\hat{m})$. Define

$$\text{(code length)}\quad \hat{n}\triangleq\|\hat{\boldsymbol{l}}\|_1, \tag{16}$$

$$\text{(user amount)}\quad \varphi\triangleq\|\hat{\boldsymbol{r}}\|_1 = \min_{\mathcal{K}\subset_K\mathcal{V}}\sum_{i\in\mathcal{K}}\hat{l}_i, \tag{17}$$

[3]This can be always achieved if we append zero rows to $\mathbf{A}$ to make the number of rows a multiple of $\hat{m}$.

$$\text{(hacker amount)} \quad \theta \triangleq \max_{\mathcal{J} \subset_J \mathcal{V}} \sum_{i \in \mathcal{J}} \hat{l}_i, \tag{18}$$

$$\text{(code dimension)} \quad \omega \triangleq \theta + \hat{m}. \tag{19}$$

Note that $\hat{n}$ is the total number of storage blocks, φ is the minimum number of encoded blocks accessible by the user, which, according to the computing requirement, equals the total amount of computing blocks, θ is the maximum number of encoded blocks accessible by the hacker, $\hat{m}$ is the number of original data blocks, and ω is the dimension of the code.

We denote the encoding functions as $\boldsymbol{f} = (f_1, \ldots, f_V)$, where $f_i : \mathbb{F}_q^{\hat{m}b \times s} \to \mathbb{F}_q^{\hat{l}_i b \times s}$. The encoding function for Cloud i, which encodes the $\hat{m}$ input blocks $\mathbf{A}_1, \ldots, \mathbf{A}_{\hat{m}}$ into $\tilde{\mathbf{A}}_i$ (with $\hat{l}_i$ blocks), is defined as follows:

$$\tilde{\mathbf{A}}_i = f_i(\mathbf{A}_1, \ldots, \mathbf{A}_{\hat{m}}) \triangleq (\mathbf{P}_i \mathbf{G}_0^T \otimes \mathbf{I}_b) \begin{bmatrix} \mathbf{R}_1 \\ \vdots \\ \mathbf{R}_\theta \\ \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{\hat{m}} \end{bmatrix}, \tag{20}$$

where $\otimes$ is the Kronecker product and $\mathbf{I}_b$ is the $b \times b$ identity matrix. $\mathbf{R}_i$, $i = 1, \ldots, \theta$, is a $b \times s$ random matrix (block) whose elements are independently distributed in $\mathbb{F}_q$. $\mathbf{G}_0$ is an $\omega \times \hat{n}$ *upper trapezoidal matrix* which can be partitioned into six submatrices as shown below:

$$\mathbf{G}_0 \triangleq \begin{bmatrix} \mathbf{I}_\theta & \mathbf{U}_{12} & \mathbf{U}_{13} \\ \mathbf{0} & \mathbf{I}_{\hat{m}} & \mathbf{U}_{23} \end{bmatrix} \triangleq \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{\eta} \end{bmatrix}, \tag{21}$$

where $\boldsymbol{\beta}$ is of size $\theta \times \hat{n}$ and $\boldsymbol{\eta}$ is of size $\hat{m} \times \hat{n}$, and $\mathbf{U}_{ij}$'s are some matrices of compatible sizes such that $\mathbf{G}_0$ has the following two properties[4]:

(1) Any $\omega \times \omega$ submatrix of $\mathbf{G}_0$ is of full rank;
(2) Any $\theta \times \theta$ submatrix of $\boldsymbol{\beta}$ is of full rank.

$\mathbf{P}_i$ is a binary matrix of size $\hat{l}_i \times \hat{n}$ which has a single "1" in each row and at most one "1" in each column. It is used to choose $\hat{l}_i$ distinct rows from the $\hat{n}$ rows of $\mathbf{G}_0^T$, and can be partitioned into two submatrices as follows:

$$\mathbf{P}_i \triangleq \left[\begin{array}{c} \boldsymbol{e}(\sum_{k=0}^{i-1} \hat{r_k} + 1) \\ \boldsymbol{e}(\sum_{k=0}^{i-1} \hat{r_k} + 2) \\ \vdots \\ \boldsymbol{e}(\sum_{k=0}^{i} \hat{r_k}) \\ \hline \boldsymbol{e}(\varphi + \sum_{k=0}^{i-1}(\hat{l}_k - \hat{r_k}) + 1) \\ \boldsymbol{e}(\varphi + \sum_{k=0}^{i-1}(\hat{l}_k - \hat{r_k}) + 2) \\ \vdots \\ \boldsymbol{e}(\varphi + \sum_{k=0}^{i}(\hat{l}_k - \hat{r_k})) \end{array}\right] \triangleq \begin{bmatrix} \mathbf{P}_i^U \\ \mathbf{P}_i^L \end{bmatrix},$$

where $\boldsymbol{e}(j)$ is the row vector corresponding to the j-th standard basis of the $\hat{n}$-dimensional vector space. $\mathbf{P}_i^U$ is the upper submatrix with $\hat{r}_i$ rows and $\mathbf{P}_i^L$ is the lower submatrix with $\hat{l}_i - \hat{r}_i$ rows. For example, in a $(3, 2, 1)$ system, let $\hat{\boldsymbol{l}} = (4, 2, 3), \hat{\boldsymbol{r}} = (2, 2, 1)$, and then we have

$$\mathbf{P}_1 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \end{bmatrix},$$

[4]Details of how to construct $\mathbf{G}_0$ are in Subsection V-D.

$$\mathbf{P}_2 = \begin{bmatrix} 0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \end{bmatrix},$$

$$\mathbf{P}_3 = \begin{bmatrix} 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \end{bmatrix},$$

where Cloud 1 chooses rows $1, 2, 6, 7$, Cloud 2 chooses rows $3, 4$, and Cloud 3 chooses rows $5, 8, 9$.

Due to the structure of $\mathbf{G}_0$, we call our code *upper trapezoidal (UT) code*. It is specially designed to facilitate fast decoding of the computation result $\mathbf{y}$, which will be discussed later. From the definition of encoding function in (20), the encoding process can also be represented in the following way:

$$\tilde{\mathbf{A}} = (\mathbf{G}^T \otimes \mathbf{I}_b) \begin{bmatrix} \mathbf{R} \\ \mathbf{A} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{A}}_1 \\ \vdots \\ \tilde{\mathbf{A}}_V \end{bmatrix}, \tag{22}$$

where $\mathbf{R}$ is the $\theta b \times s$ random matrix formed by stacking up the $\mathbf{R}_i$'s, $\mathbf{G}$ is the $\omega \times \hat{n}$ generator matrix in the form of $\mathbf{G}^T \triangleq \mathbf{P}\mathbf{G}_0^T$, and $\mathbf{P}$ is a permutation matrix of size $\hat{n}$ which is composed of $\mathbf{P}_1, \ldots, \mathbf{P}_V$ as

$$\mathbf{P} \triangleq \begin{bmatrix} \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_V \end{bmatrix}.$$

In other words, $(\hat{\boldsymbol{l}}, \hat{\boldsymbol{r}}, \hat{m})$ UT code, defined by the generator matrix $\mathbf{G}$, encodes $\hat{m}$ input blocks into $\hat{n}$ blocks, and cloud manager allocates $\hat{l}_i$ encoded blocks for Cloud i, $i = 1, \ldots, V$, according to the permutation matrix $\mathbf{P}$. The encoding operation in (22) is a matrix multiplication on a block-by-block basis. More precisely, a given element of the generator matrix is multiplied to all the elements in a data block of b rows. We call such an operation *block scalar multiplication*, which needs to be done $\hat{n}\omega$ times. Thus, the encoding complexity is $O(\hat{n}\omega bs)$.

For Cloud i, it stores $\tilde{\mathbf{A}}_i$ with $l_i/b = \hat{l}_i$ blocks and uses only r_i/b blocks to compute. The computing result $\tilde{\mathbf{y}}$ obtained by stacking the computing results of from V clouds is

$$\tilde{\mathbf{y}} = \begin{bmatrix} \tilde{\mathbf{B}}_1\mathbf{x} \\ \vdots \\ \tilde{\mathbf{B}}_V\mathbf{x} \end{bmatrix} = \mathbf{G}_s \otimes \mathbf{I}_b \begin{bmatrix} \mathbf{R}\mathbf{x} \\ \mathbf{A}\mathbf{x} \end{bmatrix}, \tag{23}$$

where $\mathbf{G}_s$ is a $\varphi \times \omega$ submatrix of $\mathbf{G}^T$ corresponding to the load vector $\boldsymbol{r}$ used. If there is no cloud outages in the system, $r_i/b = \hat{r}_i$ for all $i \in \mathcal{V}$. Hence, $\tilde{\mathbf{B}}_i$, the submatrix of $\tilde{\mathbf{A}}_i$ preserving *the first* r_i *rows*, can be written as

$$\tilde{\mathbf{B}}_i = (\mathbf{P}_i^U \mathbf{G}_0^T \otimes \mathbf{I}_b) \begin{bmatrix} \mathbf{R} \\ \mathbf{A} \end{bmatrix}.$$

And in this case $\mathbf{G}_s$ is exactly the upper $\varphi \times \omega$ submatrix of $\mathbf{G}_0^T$, which provides a good opportunity for fast decoding.

B. Code Properties

In this subsection, we will show that the proposed UT code, if $\omega \le \varphi$, satisfy the two stronger system requirements in (2) and (4). Moreover, it achieves the secrecy capacity if $\hat{\boldsymbol{l}} = \frac{\hat{m}}{K-J}\mathbf{1}_V$.

Lemma 7: An $(\hat{\boldsymbol{l}}, \hat{\boldsymbol{r}}, \hat{m})$ UT code, such that $\omega \leq \varphi$, fulfills the system requirements in (2) and (4).

Proof: First, we show that the code satisfies the computing requirement (4). The calculated outcome $\tilde{\mathbf{y}}$ received by the cloud manager, with φ blocks in total, is given by (23). Since $\varphi \geq \omega$, we can remove the bottom $\varphi - \omega$ blocks from $\mathbf{G}_s$ to obtain the square matrix $\mathbf{G}_s^*$, which is of full rank due to the property of $\mathbf{G}_0$. From the property of Kronecker product, we have

$$\operatorname{rank}(\mathbf{G}_s^* \otimes \mathbf{I}_b) = \operatorname{rank}(\mathbf{G}_s^*)\operatorname{rank}(\mathbf{I}_b) = \omega b,$$

which implies $\mathbf{G}_s^* \otimes \mathbf{I}_b$ is invertible. Removing the corresponding blocks from $\tilde{\mathbf{y}}$, $\mathbf{y}$ can be obtained by left multiplying the inverse of $\mathbf{G}_s^* \otimes \mathbf{I}_b$. Therefore, $\mathbf{y}$ is a function of $\tilde{\mathbf{y}}$, which implies (4).

Next, we consider the secrecy requirement in (2). The eavesdropper fetches arbitrarily x encoded blocks from J clouds to construct $\tilde{\mathbf{A}}_{\mathcal{J}}$, i.e.,

$$\tilde{\mathbf{A}}_{\mathcal{J}} = (\boldsymbol{\eta}' \otimes \mathbf{I}_b)\mathbf{A} + (\boldsymbol{\beta}' \otimes \mathbf{I}_b)\mathbf{R}. \tag{24}$$

where $\boldsymbol{\eta}'$ is an $x \times \hat{m}$ submatrix of $\boldsymbol{\eta}^T$ and $\boldsymbol{\beta}'$ is an $x \times \theta$ submatrix of $\boldsymbol{\beta}^T$. Since the eavesdropper has access to at most θ encoded blocks, we only need to consider the worst case when $x = \theta$ in (24), which can be expressed as

$$\tilde{\mathbf{A}}_{\mathcal{J}} = (\boldsymbol{\eta}' \otimes \mathbf{I}_b)\mathbf{A} + \mathbf{U},$$

where $\mathbf{U} = (\boldsymbol{\beta}' \otimes \mathbf{I}_b)\mathbf{R}$ is a square matrix. Due to Property 2 of the generator matrix, $\boldsymbol{\beta}'$ is of full rank, so there is a one-to-one correspondence between $\mathbf{U}$ and $\mathbf{R}$. Since the entries of $\mathbf{R}$ are independently and uniformly distributed in $\mathbb{F}_q$, so are the entries of $\mathbf{U}$. Given any realization of $\mathbf{A}$, the entries of $\tilde{\mathbf{A}}_{\mathcal{J}}$ are independently and uniformly distributed in $\mathbb{F}_q$. In other words, $\mathbf{A}$ and $\tilde{\mathbf{A}}_{\mathcal{J}}$ are mutually independent, which implies (2). □

In the following proposition, we will show that our proposed code achieves the secrecy capacity in (7), i.e., the maximum achievable rate of a (V, K, J) system.

Proposition 8: An $(\hat{\boldsymbol{l}}, \hat{\boldsymbol{r}}, \hat{m})$ UT code, such that $\hat{\boldsymbol{l}} = \frac{\hat{m}}{K-J}\mathbf{1}_V$, achieves the secrecy capacity.

Proof: If $\hat{l}_1 = \cdots = \hat{l}_V = \frac{\hat{m}}{K-J}$, then $\omega = \varphi$, and the system requirements are all fulfilled according to Lemma 7. Its achievable rate is given by $\frac{\hat{m}}{\hat{n}} = \frac{\hat{m}}{\|\hat{\boldsymbol{l}}\|_1} = \frac{K-J}{V}$, which is the secrecy capacity stated in Theorem 1. □

C. Decoding Complexities

In this subsection, we discuss the decoding scheme for computation result $\mathbf{y}$, and analyze its time complexity in terms of scalar multiplication.

The calculated outcome $\tilde{\mathbf{y}}$ received by the cloud manager, with φ blocks in total, is given by (23). Since $\varphi \geq \omega$, we can remove the bottom $\varphi - \omega$ blocks from $\mathbf{G}_s$ to obtain the square matrix $\mathbf{G}_s^*$ of size $\omega \times \omega$. Due to the property of UT code, $\mathbf{G}_s^*$ is a full-rank submatrix of generator matrix $\mathbf{G}$, implying $\mathbf{G}_s^* \otimes \mathbf{I}_b$ being invertible by Lemma 7. Removing the corresponding blocks from $\tilde{\mathbf{y}}$, $\mathbf{y}$ can be obtained by left multiplying the inverse of $\mathbf{G}_s^* \otimes \mathbf{I}_b$. Computing the inverse by Gaussian elimination requires a time complexity of $O(\omega^3)$. The operation of matrix multiplication is done block by block, requiring a time complexity of $O(\omega^2 bs)$. Therefore, the complexity of this decoding process is $O(\omega^3 + \omega^2 bs)$.

However, we can do much better when there are no cloud outages, which is the ordinary case as outages rarely occur in the cloud. The cloud manager now has the vector $\tilde{\mathbf{y}}$, which is exactly the upper sub-vector with φ blocks of $\tilde{\mathbf{A}}\mathbf{x}$, where

$$\begin{aligned}\tilde{\mathbf{A}}\mathbf{x} &= \left(\begin{bmatrix} \mathbf{I}_\theta & 0 \\ \mathbf{U}_{12}^T & \mathbf{I}_{\hat{m}} \\ \mathbf{U}_{13}^T & \mathbf{U}_{23}^T \end{bmatrix} \otimes \mathbf{I}_b \right) \begin{bmatrix} \mathbf{R}\mathbf{x} \\ \mathbf{y} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}\mathbf{x} \\ (\mathbf{U}_{12}^T \otimes \mathbf{I}_b)\mathbf{R}\mathbf{x} + \mathbf{y} \\ (\mathbf{U}_{13}^T \otimes \mathbf{I}_b)\mathbf{R}\mathbf{x} + (\mathbf{U}_{23}^T \otimes \mathbf{y}) \end{bmatrix}.\end{aligned}$$

Rewrite $\tilde{\mathbf{y}}$ as

$$\tilde{\mathbf{y}} = \begin{bmatrix} \tilde{\mathbf{y}}_1 \\ \tilde{\mathbf{y}}_2 \\ \tilde{\mathbf{y}}_3 \end{bmatrix},$$

where $\tilde{\mathbf{y}}_1 \triangleq \mathbf{R}\mathbf{x}$ is of size θb, $\tilde{\mathbf{y}}_2 \triangleq (\mathbf{U}_{12}^T \otimes \mathbf{I}_b)\mathbf{R}\mathbf{x} + \mathbf{y}$ is of size $\hat{m}b$, and $\tilde{\mathbf{y}}_3$ is of size $(\varphi - \omega)b$. Note that the computation result $\mathbf{y}$ can be decoded from $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$, and $\tilde{\mathbf{y}}_1$ has already been computed by the first E clouds, where E is the smallest integer such that $\sum_{i=1}^{E} \hat{r}_i \geq \theta$. Therefore, it is most efficient to utilize those E $(E \geq J)$ clouds to jointly compute $(\mathbf{U}_{12}^T \otimes \mathbf{I}_b)\mathbf{R}\mathbf{x}$ in parallel.

To do so, the cloud manager sends $\mathbf{D}^i$ to Cloud i to compute $(\mathbf{D}^i \otimes \mathbf{I}_b)\tilde{\mathbf{B}}_i\mathbf{x}$, for $i = 1, \ldots, E-1$. The matrix $\mathbf{D}^i$ is the submatrix of $\mathbf{U}_{12}^T$ defined as follows:

$$\mathbf{U}_{12}^T \triangleq \begin{bmatrix} \mathbf{D}^1 & \cdots & \mathbf{D}^E \end{bmatrix},$$

where $\mathbf{D}^i$ is of size $\hat{m} \times \hat{r}_i$ for $i = 1, \ldots, E-1$ and of size $\hat{m} \times (\theta - \sum_{j=0}^{E-1} \hat{r}_j)$ for $i = E$. For Cloud E, it computes $(\mathbf{D}^E \otimes \mathbf{I}_b)\tilde{\mathbf{B}}_E^{\text{Trun}}\mathbf{x}$, where $\tilde{\mathbf{B}}_E^{\text{Trun}}$ is obtained by truncating $\tilde{\mathbf{B}}_E$ with the upper $\theta - \sum_{j=0}^{E-1} \hat{r}_j$ blocks preserved. Note that $\tilde{\mathbf{B}}_i\mathbf{x}$ has already been computed by Cloud i in the previous computing stage and thus needs not to be computed again. After collecting these computational outcomes, the cloud manager adds them as $\tilde{\mathbf{y}}_E$, which is equal to $(\mathbf{U}_{12}^T \otimes \mathbf{I}_b)\mathbf{R}\mathbf{x}$. The cloud manager can then decode $\mathbf{y}$ by subtracting $\tilde{\mathbf{y}}_E$ from $\tilde{\mathbf{y}}_2$.

The computation in clouds requires block scalar multiplication for $\theta\hat{m}$ times. From Lemma 7, to satisfy system requirements, we have $\varphi \geq \theta + \hat{m}$, i.e.,

$$\begin{aligned}\theta\hat{m} \leq \theta(\varphi - \theta) &= \max_{\mathcal{J} \subset_J \mathcal{V}} \sum_{i \in \mathcal{J}} \hat{l}_i \Big(\min_{\mathcal{K} \subset_K \mathcal{V}} \sum_{i \in \mathcal{K}} \hat{l}_i - \max_{\mathcal{J} \subset_J \mathcal{V}} \sum_{i \in \mathcal{J}} \hat{l}_i \Big) \\ &\leq \max_{\mathcal{J} \subset_J \mathcal{V}} \sum_{i \in \mathcal{J}} \hat{l}_i \Big(\max_{\mathcal{K} \subset_K \mathcal{V}} \sum_{i \in \mathcal{K}} \hat{l}_i - \max_{\mathcal{J} \subset_J \mathcal{V}} \sum_{i \in \mathcal{J}} \hat{l}_i \Big).\end{aligned}$$

Hence, the decoding complexity is $O(J(K-J)) = O(KJ)$. It should be noted that the E clouds run in parallel, thereby speeding up the decoding computations by a factor of E, where $E \geq J$. While the cloud manager just needs to do addition and subtraction to decode $\mathbf{y}$, which requires no scalar multiplication at all. This is an especially desirable feature, since the clouds are supposed to have high computing power while the cloud manager has limited computing capability.

D. Instantiation of $\mathbf{G}_0$

Assigning values to the non-zero components of $\mathbf{G}_0$ is called *instantiation* of $\mathbf{G}_0$. To complete the construction of the proposed code, we need to find a qualified instantiation of $\mathbf{G}_0$ such that the two properties stated in subsection A are satisfied. Such an instantiation is said to be *feasible*. By utilizing random codes, a feasible instantiation can be easily found as long as the field size is sufficiently large. This result is formally stated in the following proposition, whose proof is in the Appendix.

Proposition 9: Let $\mathbf{U}_{ij}$'s be random matrices with entries uniformly and independently picked from finite field $\mathbb{F}_q$. Such an instantiation of $\mathbf{G}_0$ is feasible with probability 1 when the field size, q, goes to infinity.

Proposition 9 tells us that a feasible instantiation exists and can be obtained randomly when the field size is large enough. On the other hand, it is not clear how large the field should be. In what follows, we are going to show that a feasible instantiation can be obtained by our proposed deterministic algorithm. Moreover, a bound on the required field size is derived.

Denote the i-th column of $\mathbf{G}_0$ by a column ω-vector $\boldsymbol{f}_i$, which can be written as

$$\boldsymbol{f}_i \triangleq \begin{bmatrix} \boldsymbol{f}_i^* \\ \boldsymbol{f}_i' \end{bmatrix},$$

where $\boldsymbol{f}_i^*$ and $\boldsymbol{f}_i'$ are column subvectors of size θ and $\hat{m}$, respectively. Then we can rewrite $\mathbf{G}_0$, of size $\hat{n} \times \omega$, as

$$\mathbf{G}_0 = \begin{bmatrix} \mathbf{I}_\theta & \mathbf{U}_{12} & \mathbf{U}_{13} \\ \mathbf{0} & \mathbf{I}_{\hat{m}} & \mathbf{U}_{23} \end{bmatrix} = [\boldsymbol{f}_1 \ \cdots \ \boldsymbol{f}_{\hat{n}}] = \begin{bmatrix} \boldsymbol{f}_1^* & \cdots & \boldsymbol{f}_{\hat{n}}^* \\ \boldsymbol{f}_1' & \cdots & \boldsymbol{f}_{\hat{n}}' \end{bmatrix}.$$

$\mathbf{U}_{12}$ is constructed by a Cauchy matrix, whose (i,j)-th element is given by $1/(x_i - y_j)$ with distinct elements $x_1, x_2, \ldots, x_{\hat{m}}, y_1, y_2, \ldots y_\theta$ picked from $\mathbb{F}_q$. Then the submatrix formed by collecting $\boldsymbol{f}_1, \ldots \boldsymbol{f}_\omega$ satisfies the requried two properties, because any square submatrix of a Cauchy matrix is nonsingular over any field [42, p.323]. It remains to find qualified $\boldsymbol{f}_{\omega+1}, \ldots \boldsymbol{f}_{\hat{n}}$ such that the two properties required of $\mathbf{G}_0$ are satisfied, which yields a feasible instantiation.

Given a set of index $\xi \subset \{1, 2, \ldots, \hat{n}\}$, the vector space, denoted V_ξ, is spanned by the column vectors $\boldsymbol{f}_i$ where $i \in \xi$. Similarly, V_ψ^* is defined as the vector space spanned by subvectors $\boldsymbol{f}_i^*$ where $i \in \psi \subset \{1, 2, \ldots, \hat{n}\}$. Our construction method is stated in Algorithm 1, which follows the same methodology of constructing generic network codes in [40], [41]. The correctness of Algorithm 1 and the required field size are proved below:

Proposition 10: If the field size satisfies

$$q > \tbinom{\hat{n}-1}{\omega-1} + \tbinom{\hat{n}-1}{\theta-1} + \omega, \tag{25}$$

then Algorithm 1 can yield a feasible instantiation of $\mathbf{G}_0$.

Proof: Given (25), we will first show that such a vector $\boldsymbol{x}$ in Step 12 always exists. Since ξ is a collection of vectors $\{\boldsymbol{f}_i : i \in \xi\}$ with $|\xi| = \omega - 1$, so $\dim(V_\xi) \le \omega - 1$, then we have $|V_\xi| \le q^{\omega-1}$. Similarly, $\dim(V_\psi^*) \le \theta - 1$ implies $|V_\psi^*| \le q^{\theta-1}$. The vectors in V_ξ and subvectors in V_ψ^* should not be chosen. The number of such collections ξ and ψ are at most $\binom{\hat{n}-1}{\omega-1}$ and $\binom{\hat{n}-1}{\theta-1}$, respectively. Moreover, the vectors with zero elements should also be excluded. The number of such vectors is $q^\omega - (q-1)^\omega < \omega q^{\omega-1}$. Therefore, the total number of vectors in $\mathbb{F}_q^\omega$ that should not be picked is bounded by

Algorithm 1 Assign Column Vectors for Matrix $\mathbf{G}_0$

```
Input: $\theta, \hat{m}, \hat{n}$
Output: $\{\boldsymbol{f}_i = \begin{bmatrix} \boldsymbol{f}_i^* \\ \boldsymbol{f}_i' \end{bmatrix}, i = 1, \ldots, \hat{n}\}$
1: for $i := 1, 2, \ldots, \theta$ do
2:     $\boldsymbol{f}_i^* :=$ the $i$-th natural basis of the $\theta$-dim vector space;
3:     $\boldsymbol{f}_i' := \mathbf{0}$;
4: end for
5: pick distinct $x_1, x_2, \ldots, x_{\hat{m}}, y_1, y_2, \ldots y_\theta$ from $\mathbb{F}_q$.
6: for $i := \theta+1, \theta+2, \ldots, \omega$ do
7:     $\boldsymbol{f}_i^* :=$ the $\theta$-vector whose $j$-th element is $1/(x_i - y_j)$;
8:     $\boldsymbol{f}_i' :=$ the $(i-\theta)$-th natural basis of the $\hat{m}$-dim vector space;
9: end for
10: $\mathcal{I} := \{1, \ldots, \omega\}$;
11: for $i := \omega+1, \omega+2, \ldots, \hat{n}$ do
12:     pick any vector with non-zero elements $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_i^* \\ \boldsymbol{x}_i' \end{bmatrix} \in \mathbb{F}_q^\omega$ such that $\boldsymbol{x} \notin V_\xi$ and $\boldsymbol{x}^* \notin V_\psi^*$, for all $\xi \subset_{\omega-1} \mathcal{I}$ and $\psi \subset_{\theta-1} \mathcal{I}$ such that $\{\boldsymbol{f}_e : e \in \xi\}$ are linearly independent and $\{\boldsymbol{f}_e^* : e \in \psi\}$ are linearly independent;
13:     $\boldsymbol{f}_i := \boldsymbol{x}$;
14:     $\mathcal{I} := \mathcal{I} \cup \{i\}$;
15: end for
```

$$\begin{aligned}
&|\cup_\xi V_\xi| + q^{\omega-\theta}|\cup_\psi V_\psi^*| + q^\omega - (q-1)^\omega \\
<&\sum_\xi |V_\xi| + q^{\omega-\theta}\sum_\psi |V_\psi^*| + \omega q^{\omega-1} \\
\le&\sum_\xi q^{\omega-1} + q^{\omega-\theta}\sum_\psi q^{\theta-1} + \omega q^{\omega-1} \\
\le&\tbinom{\hat{n}-1}{\omega-1}q^{\omega-1} + \tbinom{\hat{n}-1}{\theta-1}q^{\omega-1} + \omega q^{\omega-1} \\
=&\left[\tbinom{\hat{n}-1}{\omega-1} + \tbinom{\hat{n}-1}{\theta-1} + \omega\right]q^{\omega-1} \\
\overset{(a)}{<}& q^\omega = |\mathbb{F}_q^\omega|,
\end{aligned}$$

where (a) follows from (25). Therefore, such a vector $\boldsymbol{x}$ in the proposed algorithm can always be picked. As a result, Algorithm 1 will eventually halt.

We claim that the resultant matrix $\mathbf{G}_0$ is a feasible instantiation. Consider the left submatrix which consists of the first m columns of $\mathbf{G}_0$, where $m \ge \omega$. As discussed above, when $m = \omega$, the claim is true. Assume the claim is true for $m-1$ with $\omega+1 \le m \le \hat{n}$. Define $\mathcal{I} \triangleq \{1, \ldots, m\}$. Consider two arbitrary sets $\xi \subset_{\omega-1} \mathcal{I} \setminus \{m\}$ and $\psi \subset_{\theta-1} \mathcal{I} \setminus \{m\}$. By induction hypothesis, both $\{\boldsymbol{f}_e : e \in \xi\}$ and $\{\boldsymbol{f}_e^* : e \in \psi\}$ are linearly independent sets. Following Step 12, assign to $\boldsymbol{f}_m$ any vector with non-zero elements $\boldsymbol{x} = [\boldsymbol{x}' \ \boldsymbol{x}^*]^T \in \mathbb{F}_q^\omega$. As for all ξ, $\boldsymbol{x} \notin V_\xi$ implies that $\{\boldsymbol{f}_e : e \in \xi \cup \{m\}\}$ are linearly independent, so any ω vectors of $\{\boldsymbol{f}_i : i \in \mathcal{I}\}$ are linearly independent. Likewise, any θ vectors of $\{\boldsymbol{f}_i^* : i \in \mathcal{I}\}$ are linearly independent due to $\boldsymbol{x}^* \notin V_\psi^*$. Hence, by mathematical induction, the claim is proved. □

VI. Storage and Computing Optimization

In this section, given a feasible storage budget, we first derive a lower bound on the minimum computation time by elementary manipulations of Shannon's information measures, and show that UT code can achieve that bound for sufficiently large n. The fundamental tradeoff curve between storage budget and computation time is characterized and is shown to be piecewise linear. Moreover, by the notion of majorization, it is proved that the more uneven the computing rates of individual clouds, the longer the computation time.

A. The Optimal Tradeoff

In the (V, K, J) system, we aim to perform secure distributed matrix multiplication over V clouds in the presence of $V-K$ cloud outages and J colluding clouds. The confidential matrix $\mathbf{A} \in \mathbb{F}_q^{m\times s}$ is encoded into $\tilde{\mathbf{A}} \in \mathbb{F}_q^{n\times s}$ with code rate m/n. Given storage allocation vector $\boldsymbol{l}$, where $\|\boldsymbol{l}\|_1 = n$, we allocate the encoded matrix to V clouds for storage, and then perform distributed computing according to the load vector $\boldsymbol{r}$. Without loss of generality, we assume the computing service rates of clouds are in descending order: $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_V$. The budget of normalized storage cost on multiple clouds is S, i.e.,

$$n = \sum_{i\in\mathcal{V}} l_i \leq mS. \tag{26}$$

Since $\boldsymbol{l}$ and $\boldsymbol{r}$ naturally grow with the number of data records, m, we normalize both of them by m. With slight abuse of notation, we use the same symbols but interpret them as the storage amount and computing amount *per data record*. We are particularly interested in the asymptotic case where m is large, which is relevant to big data applications. While those variables are rational numbers by nature, it is justifiable to make a relaxation, treating them as real numbers.

We can then obtain a lower bound on the minimum computation time by solving the following problem:

$$T_{LB} \triangleq \min_{\boldsymbol{l},\boldsymbol{r}\in\mathbb{R}^V} \{\max_{i\in\mathcal{V}} \frac{r_i}{\mu_i}\} \tag{27}$$

subject to

$$\sum_{i\in\mathcal{V}} l_i \leq S \tag{28}$$

$$\min_{\mathcal{K}\subset_K\mathcal{V}} \sum_{i\in\mathcal{K}} l_i - \max_{\mathcal{J}\subset_J\mathcal{V}} \sum_{i\in\mathcal{J}} l_i \geq 1 \tag{29}$$

$$0 \leq r_i \leq l_i,\ \forall i \in \mathcal{V} \tag{30}$$

$$\sum_{i\in\mathcal{V}} r_i = \min_{\mathcal{K}\subset_K\mathcal{V}} \sum_{i\in\mathcal{K}} l_i, \tag{31}$$

where (29) comes from the necessary condition (8) for the system requirements. Since (28), (30) and (31) are also necessary conditions to be met, if we minimize the computation time subject to these constraints, the result so obtained becomes a lower bound for the computation time.

Note the lower bound T_{LB} is well defined if and only if the optimization problem (27) is feasible. The feasibility condition is shown in the following lemma, with proof in the Appendix. We can see $S_0 = 1/C$, implying that the optimization problem is feasible if and only if the code rate applied by the system does not exceed the secrecy capacity. That is, secrecy capacity is equivalent to the feasibility condition for the optimization problem.

Lemma 11: The optimization problem (27) is feasible if and only if $S \geq S_0 \triangleq \frac{V}{K-J}$. In particular, if $S = S_0$, the storage allocation must be equal, i.e., $l_1 = \cdots = l_V = \frac{1}{K-J} \triangleq l_0$.

Lemma 12: Given a feasible problem instance, there always exists an optimal allocation vector $\boldsymbol{l}^*$ at which $l_1^* \geq l_2^* \geq \cdots \geq l_V^*$ and equality holds in (29).

The proof is in the Appendix. Note that a feasible problem instance may not have a unique solution. For example, the slowest clouds can exchange the values of their storage allocations l_i's without increasing the minimum computation time as long as the constraints $r_i \leq l_i$ are fulfilled.

Lemma 13: $T^*(S) = T_{LB}$ for any feasible storage budget S.

Proof: We claim that T_{LB} can be achieved by an $(\hat{\boldsymbol{l}}^*, \hat{\boldsymbol{r}}^*, \hat{m})$ UT code. Let $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ be an optimal solution to the optimization problem (27), and $\boldsymbol{l}^*$ satisfies the conditions in Lemma 12. For $i \in \mathcal{K}_0 \triangleq \{V-K+1, \ldots, V\}$, let $\hat{l}_i^* = \lfloor \hat{m} l_i^* \rfloor$ and $\hat{r}_i^* = \lfloor \hat{m} r_i^* \rfloor$. For $i \in \mathcal{V} \setminus \mathcal{K}_0$, let $\hat{l}_i^* = \lceil \hat{m} l_i^* \rceil$ and $\hat{r}_i^* = \lceil \hat{m} r_i^* \rceil$. This setting of $\hat{\boldsymbol{l}}^*$ ensures that $\omega \geq \varphi$, which is required for the code to fulfill all system requirements as shown in Lemma 7. The normalized storage amount after encoding is $\hat{S} = \frac{\|\hat{\boldsymbol{l}}^*\|_1}{\hat{m}} = \frac{\hat{n}}{\hat{m}}$. It can be seen that $\hat{S} \to S$ as $n \to \infty$, since $\frac{\hat{l}_i^*}{\hat{m}} \to l_i^*$ as $n \to \infty$.

Next, we may need to adjust $\hat{\boldsymbol{r}}^*$ such that the following two conditions hold: (a) $\hat{\boldsymbol{r}}^* \leq \hat{\boldsymbol{l}}^*$ and (b) the condition in (17). It is easy to check that condition (a) holds under the current setting. If $\|\hat{\boldsymbol{r}}^*\|_1 > \sum_{i\in\mathcal{K}_0} \hat{l}_i^*$, we can reduce one or more non-zero components of $\hat{\boldsymbol{r}}^*$ such that equality holds. Otherwise, if $\|\hat{\boldsymbol{r}}^*\|_1 < \sum_{i\in\mathcal{K}_0} \hat{l}_i^*$, we can increase r_i by 1 for one or more $i \in \mathcal{K}_0$ such that equality holds. After the adjustment of $\hat{\boldsymbol{r}}^*$, for both cases, we must have $\hat{r}_i^* \leq \lceil \hat{m} r_i^* \rceil$ for all $i \in \mathcal{V}$. Moreover, both (a) and (b) hold. Given any $\zeta > 0$, when $\hat{m} \to \infty$, we have $\frac{\hat{r}_i}{\hat{m}} \leq r_i^* + \epsilon$, where $\epsilon \triangleq \zeta \min_{i\in\mathcal{V}} \mu_i$. Hence, the computation time T achieved by the UT code satisfies

$$T_{LB} \leq T = \max_{i\in\mathcal{V}} \frac{\hat{r}_i^*}{\hat{m}\mu_i} \leq \max_{i\in\mathcal{V}} \frac{r_i^* + \epsilon}{\mu_i} \leq T_{LB} + \zeta.$$

Hence, (S, T_{LB}) is feasible, implying $T^*(S) = T_{LB}$. □

Lemma 13 implies that the minimum computation time can be obtained by solving the optimization problem in (27), which can be be simplified as follows by Lemma 12:

$$\min_{\boldsymbol{l},\boldsymbol{r}\in\mathbb{R}^V} T \tag{32}$$

subject to

$$\sum_{i\in\mathcal{V}} l_i \leq S \tag{33}$$

$$l_1 \geq l_2 \geq \cdots \geq l_V \tag{34}$$

$$\sum_{i=V-K+1}^{V} l_i - \sum_{i=1}^{J} l_i = 1 \tag{35}$$

$$0 \leq r_i \leq l_i,\ \forall i \in \mathcal{V} \tag{36}$$

$$T \geq \frac{r_i}{\mu_i},\ \forall i \in \mathcal{V} \tag{37}$$

$$\sum_{i\in\mathcal{V}} r_i = \sum_{i=V-K+1}^{V} l_i \tag{38}$$

B. Bottleneck Clouds

In this subsection, we introduce a useful concept called *bottleneck* clouds, which denote the clouds that are the last ones to finish computing. The number of non-bottleneck clouds at *minimum* storage budget S_0, which we are going to denote by Q, turns out to be an important value that affects the overall system performance.

Definition 7: Cloud i is said to be a *bottleneck* if $r_i^* = \mu_i T^*$. The index set of bottleneck clouds for budget S is

$$\mathcal{B}_{\Delta s} \triangleq \{i \in \mathcal{V} : r_i^* = \mu_i T^*\},$$

where $\Delta s \triangleq S - S_0 \geq 0$. Accordingly, others are called *non-bottleneck* clouds, whose index set is the complement of $\mathcal{B}_{\Delta s}$, i.e.,

$$\bar{\mathcal{B}}_{\Delta s} \triangleq \mathcal{V} \setminus \mathcal{B}_{\Delta s} = \{i \in \mathcal{V} : r_i^* < \mu_i T^*\}.$$

Let $x(S) \triangleq |\bar{\mathcal{B}}_{\Delta s}|$ be the number of non-bottleneck clouds with storage budget S. (When there is no ambiguity, we may simply write it as x.)

First, we consider the scenario with minimum storage budget, i.e., $\Delta s = 0$. Denote the corresponding minimum completion time by T_0, achieved by $\boldsymbol{l}_0 \triangleq (l_1^0, l_2^0, \ldots, l_V^0)$ and $\boldsymbol{r}_0 \triangleq (r_1^0, r_2^0, \ldots, r_V^0)$. From Lemma 11, the allocation vector must be $\boldsymbol{l}_0 = l_0 \mathbf{1}_V$. The total computation load in (38) then becomes

$$\sum_{i\in\mathcal{V}} r_i = K l_0. \tag{39}$$

Recall $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_V$. For notation simplicity, let μ_0 be a sufficiently large number such that $\frac{l_0}{\mu_0} < T_0$. Then there exists an integer $Q \in \{0, 1, \ldots, V\}$ such that

$$\frac{l_0}{\mu_0} \leq \frac{l_0}{\mu_1} \leq \cdots \leq \frac{l_0}{\mu_Q} < T_0 \leq \frac{l_0}{\mu_{Q+1}} \leq \cdots \leq \frac{l_0}{\mu_V}. \tag{40}$$

Note that Q can be uniquely determined as T_0 is well defined. The following result shows that there are Q non-bottleneck clouds if $\Delta s = 0$.

Lemma 14: With minimum storage budget S_0, the optimal solution is given by $x(S_0) = Q$,

$$T_0 = \frac{(K-Q)l_0}{\sum_{i\in\mathcal{B}_0}\mu_i}, \text{ and} \tag{41}$$

$$r_i^0 = \begin{cases} l_0 & i \in \bar{\mathcal{B}}_0 \\ \mu_i T_0 & i \in \mathcal{B}_0 \end{cases}, \tag{42}$$

where $\bar{\mathcal{B}}_0 = \{1, 2, \ldots, Q\}$ and $\mathcal{B}_0 = \{Q+1, Q+2, \ldots, V\}$. Furthermore, Q is the minimum integer such that

$$Q + \frac{1}{\mu_{Q+1}} \sum_{i\in\mathcal{B}_0} \mu_i \geq K. \tag{43}$$

Proof: Define $\mathcal{I} \triangleq \{1, 2, \ldots, Q\}$. By the definition of Q, $r_i^0 \leq l_i < \mu_i T_0, \forall i \in \mathcal{I}$. Then $\mathcal{B}_0 \cap \mathcal{I} = \varnothing$, i.e., $\mathcal{B}_0 \subseteq \mathcal{V} \setminus \mathcal{I}$.

Suppose $\mathcal{B}_0 \subsetneq \mathcal{V} \setminus \mathcal{I}$. Pick any $j \in (\mathcal{V} \setminus \mathcal{I}) \setminus \mathcal{B}_0$, and we set r_i^0 as $r_i^0 - \eta$ for all $i \in \mathcal{B}_0$ and r_j^0 as $r_j^0 + |\mathcal{B}_0|\eta$, where $\eta > 0$ is arbitrarily small. Then the resultant computation time $T' < T_0$, which contradicts with T_0 being optimal. Hence, $\mathcal{B}_0 = \mathcal{V} \setminus \mathcal{I}$, $\bar{\mathcal{B}}_0 = \mathcal{I}$, and $x = Q$.

Suppose $r_i^0 < l_0$ for some $i \in \bar{\mathcal{B}}_0$. If we set r_i^0 as $r_i^0 + \epsilon$ and r_j^0 as $r_j^0 - \frac{\epsilon}{|\mathcal{B}_0|}\ \forall j \in \mathcal{B}_0$, where $\epsilon > 0$ is arbitrarily small, then the resultant computation time $T'' < T_0$, which contradicts with T_0 being optimal. Therefore, $\forall i \in \bar{\mathcal{B}}_0$, $r_i^0 = l_0$. Hence, by (39), we have

$$T_0 = \frac{\sum_{i\in\mathcal{B}_0} r_i^0}{\sum_{i\in\mathcal{B}_0}\mu_i} = \frac{\sum_{i\in\mathcal{V}} r_i^0 - \sum_{i\in\bar{\mathcal{B}}_0} r_i^0}{\sum_{i\in\mathcal{B}_0}\mu_i} = \frac{Kl_0 - Ql_0}{\sum_{i\in\mathcal{B}_0}\mu_i},$$

which, together with (40), implies that Q is the minimum integer that satisfies (43). □

Next, we will discuss the case where $\Delta s > 0$. Define $\Delta l_i \triangleq l_i - l_0$ for $i \in \mathcal{V}$, which is not necessarily non-negative. To ensure feasibility, Δl_i's have to satisfy (33), (34) (35), and (36), which is equivalent to

$$0 \leq \sum_{i\in\mathcal{V}} \Delta l_i \leq \Delta s, \tag{44}$$

$$\Delta l_1 \geq \Delta l_2 \geq \cdots \geq \Delta l_V \geq -l_0, \tag{45}$$

$$\sum_{i=V-K+1}^{V} \Delta l_i = \sum_{i=1}^{J} \Delta l_i, \tag{46}$$

where the last equality follows from the definition of l_0 in Lemma 11.

Lemma 15: $\Delta l_1 \geq \Delta l_2 \geq \cdots \geq \Delta l_J \geq 0$.

Proof: Assume $\Delta l_J < 0$. According to (45) and (46), we have $\sum_{i=J+1}^{V} \Delta l_i = \sum_{i=1}^{V-K} \Delta l_i < 0$. The smallest term in the second summation, Δl_{V-K}, must be negative, which by (45) again, implies $\sum_{i=V-K+1}^{V} \Delta l_i < 0$. Therefore, $\sum_{i\in\mathcal{V}} \Delta l_i < 0$, which contradicts with (44), so we must have $\Delta l_J \geq 0$. Therefore, $\Delta l_1 \geq \Delta l_2 \geq \cdots \geq \Delta l_J \geq 0$. □

Let the overall computing time T be achieved by $\boldsymbol{l} \triangleq (l_0 + \Delta l_1, l_0 + \Delta l_2, \ldots, l_0 + \Delta l_V)$ and $\boldsymbol{r} \triangleq (r_1, r_2, \ldots, r_V)$. Since $\mu_i T \geq r_i$ for all $i \in \mathcal{V}$ in general and all $i \in \mathcal{B}_0$ in particular, we have

$$\begin{aligned} T &\geq \frac{\sum\limits_{i\in\mathcal{B}_0} r_i}{\sum\limits_{i\in\mathcal{B}_0}\mu_i} = \frac{\sum\limits_{i\in\mathcal{V}} r_i - \sum\limits_{i\in\bar{\mathcal{B}}_0} r_i}{\sum\limits_{i\in\mathcal{B}_0}\mu_i} \\ &\overset{(a)}{\geq} \frac{\sum\limits_{i=V-K+1}^{V} l_i - \sum\limits_{i\in\bar{\mathcal{B}}_0} l_i}{\sum\limits_{i\in\mathcal{B}_0}\mu_i} \\ &= \frac{(K-Q)l_0 + \sum\limits_{i=V-K+1}^{V} \Delta l_i - \sum\limits_{i\in\bar{\mathcal{B}}_0} \Delta l_i}{\sum\limits_{i\in\mathcal{B}_0}\mu_i} \\ &\overset{(b)}{=} \frac{(K-Q)l_0 + \sum\limits_{i=1}^{J} \Delta l_i - \sum\limits_{i=1}^{Q} \Delta l_i}{\sum\limits_{i\in\mathcal{B}_0}\mu_i}, \end{aligned} \tag{47}$$

where (a) comes from (36) and (38), and (b) comes from (46). Note that T equals the expression in (47) if and only if $\mathcal{B}_{\Delta s} \supseteq \mathcal{B}_0$ and $r_i^* = l_i^*$ for all $i \in \bar{\mathcal{B}}_0$.

C. Proof of Theorem 2

In this subsection, we prove Theorem 2 by solving the optimization problem in (32) and deriving explicit forms for the minimum computation time $T^*(S)$, showing that T^* is a piecewise-linear and decreasing function of storage budget S. Specifically, depending on the value of Q, the number of non-bottleneck clouds when $\Delta s = 0$, we divide the proof into two cases: $Q \leq J$ and $Q > J$. In the first case, T^* remains unchanged as the increase of S, which is proved in Proposition 16. The second case is more involved, which can be further divided into two sub-cases according to the system parameters of V, K, and J, and in this case T^* is proved to be piecewise-linear, decreasing function of S by Proposition 18.

1) First Case: $Q \leq J$:

Proposition 16: If $Q \leq J$, the optimal computation time is $T^* = T_0$ for any $S \geq S_0$, which can be achieved by $\boldsymbol{l}^* = \boldsymbol{l}_0$ and $\boldsymbol{r}^* = \boldsymbol{r}_0$.

Proof: By Lemmas 14 and 15, we can re-write (47) as

$$T - T_0 \geq \frac{\sum_{i=Q+1}^{J} \Delta l_i}{\sum_{i \in \mathcal{B}_0} \mu_i} \geq 0, \tag{48}$$

which means the computation time cannot be less than T_0 even if there is more storage budget than S_0. Therefore, the optimal computation time is $T^* = T_0$, which can be achieved by $\boldsymbol{l}^* = \boldsymbol{l}_0$ and $\boldsymbol{r}^* = \boldsymbol{r}_0$. □

2) Second Case: $Q > J$: Next, we consider the case where $Q > J$. Let there be π distinct computing rates for the clouds indexed by $\mathcal{X} \triangleq \{J, \ldots, Q\}$, where $1 \leq \pi \leq Q - J + 1$. We partition them into different subsets, putting those clouds with the same rate μ_i into a set.

Definition 8: Let $\mathcal{D}_1 \triangleq \{i \in \mathcal{X} : \mu_i = \mu_Q\} = \{Q - q_1 + 1, \ldots, Q\}$, where $q_1 = |\mathcal{D}_1|$. If $J \notin \mathcal{D}_1$, define $\mathcal{D}_2 \triangleq \{i \in \mathcal{X} : \mu_i = \mu_{Q-q_1}\} = \{Q - q_2 + 1, \ldots, Q - q_1\}$, where $q_2 = |\mathcal{D}_1| + |\mathcal{D}_2|$. If $J \notin \mathcal{D}_2$, continue defining $\mathcal{D}_3$, $\mathcal{D}_4$, ... until $J \in \mathcal{D}_\pi$, where $\mathcal{D}_\pi \triangleq \{i \in \mathcal{X} : \mu_i = \mu_J\}$. Furthermore, for $k = 1, 2, \ldots, \pi$, define $\mu_{(k)} \triangleq \mu_i$ for any $i \in \mathcal{D}_k$.

As will be shown in Proposition 18, when the storage budget S increases, the number of non-bottleneck clouds, $x(S)$, will decrease monotonically from Q to $Q - q_1$, ..., $Q - q_{\pi-1}$ and finally to J. This is illustrated in Fig. 3. The number of descents of x is denoted by

$$\Pi \triangleq \begin{cases} \pi - 1 & \text{if } |\mathcal{D}_\pi| = 1 \\ \pi & \text{otherwise} \end{cases}.$$

Based on the lower bound on T established in Lemma 17, we find the optimal computing time T^* as well as the corresponding optimal allocation $\boldsymbol{l}^*$ and $\boldsymbol{r}^*$ in Proposition 18. Their proofs are placed in the appendices.

Lemma 17: If $Q > J$, then

$$T \geq \frac{(K - Q)l_0 - (Q - J)\lambda\Delta s}{\sum_{i \in \mathcal{B}_0} \mu_i} \triangleq T_Q(\Delta s), \tag{49}$$

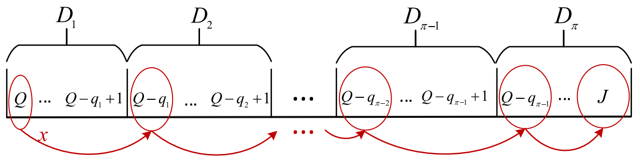

Fig. 3. As the storage budget S increases, the number of non-bottleneck clouds, x, drops from Q to $Q - q_1$, $Q - q_2$, ..., $Q - q_{\pi-1}$, and then to J.

where

$$\lambda \triangleq \begin{cases} \frac{1}{\frac{J+V-K}{K}} & \text{if } Q > V - K. \\ \frac{K}{KQ+JV-JQ} & \text{otherwise.} \end{cases} \tag{50}$$

Moreover, if $Q > V - K$, $T_Q(\Delta s)$ can be achieved by a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ if and only if it satisfies

$$\mathcal{B}_{\Delta s} \supseteq \mathcal{B}_0 \text{ and } r_i^* = l_i^* = l_0 + \lambda\Delta s \; \forall i \in \bar{\mathcal{B}}_0. \tag{51}$$

Otherwise, $T_Q(\Delta s)$ can be achieved by a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ if and only if it satisfies

$$\begin{aligned} &\mathcal{B}_{\Delta s} \supseteq \mathcal{B}_0, r_i^* = l_i^* = l_0 + \lambda\Delta s \; \forall i \in \bar{\mathcal{B}}_0, \text{ and} \\ &\Delta l_{Q+1} = \cdots = \Delta l_V = \frac{J}{K}\lambda\Delta s. \end{aligned} \tag{52}$$

Proposition 18: Denote $q_0 \triangleq 0$. If $Q > J$, for any $S \geq S_0$,

$$x(S) = \begin{cases} Q - q_0 & S_0 \leq S < S_1^{(0)} \\ Q - q_1 & S_1^{(0)} \leq S < S_2^{(0)} \\ \vdots & \\ Q - q_{\pi-1} & S_{\Pi-1}^{(0)} \leq S < S_\Pi^{(0)} \\ J & S \geq S_\Pi^{(0)} \end{cases}. \tag{53}$$

(i) If $J \geq V - K$: for $j = 1, 2, \ldots, \Pi$,

$$S_j^{(0)} = S_0 + \frac{\mu_{y_j} l_0 (K - y_j) - l_0 \sum_{i=y_j+1}^{V} \mu_i}{\lambda \sum_{i=y_j+1}^{V} \mu_i + \lambda \mu_{y_j}(y_j - J)}, \tag{54}$$

where $y_j \triangleq Q - q_{j-1}$. When $S_0 \leq S \leq S_\Pi^{(0)}$, the optimal computing time is

$$T^* = \frac{(K - x)l_0 - (x - J)\lambda\Delta s}{\sum_{i=x+1}^{V} \mu_i}, \tag{55}$$

which can be achieved by a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$

$$r_i^* = \begin{cases} l_0 + \lambda\Delta s = l_i^* & i \in \{1, 2, \ldots, x\} \\ \mu_i T^* & i \in \{x+1, x+2, \ldots, V\} \end{cases}, \tag{56}$$

$$l_i^* = r_i^* + \delta_i, \; i \in \{x+1, x+2, \ldots, V\}, \tag{57}$$

where δ_i's can be any non-negative numbers satisfying

$$\sum_{i=x+1}^{V} \delta_i = (V - K)(l_0 + \lambda\Delta s), \tag{58}$$

and

$$l_0 + \lambda\Delta s \geq r_{x+1}^* + \delta_{x+1} \geq \cdots \geq r_V^* + \delta_V. \tag{59}$$

When $S > S_\Pi^{(0)}$, T^*, r_i^*'s and l_i^*'s are equal to the case where $S = S_\Pi^{(0)}$.

(ii) If $J < V - K$, for $j = 1, 2, \ldots, j_{\max}$, where $j_{\max}$ is the largest index satisfying $Q - q_{j-1} \geq V - K$, the value of

$S_j^{(0)}$ can be obtained by (54). For $S_0 \le S < S_{j_{\max}}^{(0)}$, T^*, $\boldsymbol{r}^*$ and $\boldsymbol{l}^*$ can be obtained by the same formulas in case (i).

For $j > j_{\max}$ and $S > S_{j_{\max}}^{(0)}$, $S_j^{(0)}$, T^*, $\boldsymbol{r}^*$ and $\boldsymbol{l}^*$ can be obtained by Algorithm 2, where T^* can be expressed as

$$T^* = \frac{x + xI - (x-J)S}{(x+JI)\sum\limits_{i=x+1}^{V}\mu_i - (x-J)\sum\limits_{i\in\mathcal{I}}\mu_i}, \tag{60}$$

where $I \triangleq \frac{V-x-|\mathcal{I}|}{K}$, $\mathcal{I} \triangleq \cup_{i\in\mathcal{N}}\mathcal{D}_i$, and $\mathcal{N} \subseteq \{1,2,\ldots,\pi\}$ depends on S and can be determined by Algorithm 2.

Proposition 18 reveals that T^* is a piecewise linear function of the storage budget S. For the first case where $J \ge V-K$, it decreases linearly at rate k, where

$$k = \frac{(x-J)\lambda}{\sum\limits_{i=x+1}^{V}\mu_i}, \tag{61}$$

with Π turning points occurring at $S = S_i^{(0)}$ for $i = 1,2,\ldots,\Pi$. Each turning point indicates a change of the number of non-bottleneck clouds, $x(S)$, which is also reflected in the definition of $\mathcal{D}_i$ for $i = 1,2,\ldots,\pi$.

The second case where $J < V-K$ is more involved. As for the first case, there are Π main turning points, which indicate the change of the value of $x(S)$. On the other hand, for $j > j_{\max}$, there may be additional turning points within an interval $[S_j^{(0)}, S_{j+1}^{(0)})$, denoted by $S_{j+1}^{(z)}$ for $z \ge 1$. The objective of Algorithm 2 is to find all the additional turning points as well as the main turning points. Since those additional turning points could possibly exist only if $x(S) < V-K$, the second case is the same as case 1 when $x(S) \ge V-K$. It should be noted that in both cases, the decreasing rate of T^* drops with the increase of S, implying a convexity property. When $x(S)$ drops to J, T^* becomes a constant function of S. Hence, the storage budget of $S_\Pi^{(0)}$ is enough to achieve the minimum computing time.

To facilitate the presentation of Algorithm 2, we define a variable k_{start}, which indicates the beginning of the searching process. If $Q > V-K$, then there exists i such that $V-K \in \mathcal{D}_i$ and we let $k_{start} = i-1$. Otherwise, let $k_{start} = 0$. Furthermore, the following definition is required for the description of Algorithm 2:

Definition 9: Given $\mathcal{N} \subseteq \{1,2,\ldots,\pi\}$ and $x \in \{1,2,\ldots,V\}$, let $\mathcal{L}(\mathcal{N},x)$ be the following set of linear equations:

$$l_i^* = \begin{cases} l_1^*, & i\in\{1,2,\ldots,x\} \\ \mu_i T^*, & i\in\mathcal{I}\triangleq\cup_{i\in\mathcal{N}}\mathcal{D}_i \\ l_V^*, & i\in\{x+1,x+2,\ldots,V\}\setminus\mathcal{I} \end{cases},$$

$$r_i^* = \begin{cases} l_1^* & i\in\{1,2,\ldots,x\} \\ \mu_i T^* & i\in\{x+1,x+2,\ldots,V\} \end{cases},$$

$$\begin{cases} \sum_{i=1}^{V} l_i^* = S \\ \sum_{i=V-K+1}^{V} l_i^* - \sum_{i=1}^{J} l_i^* = 1 \\ \sum_{i=V-K+1}^{V} l_i^* = \sum_{i=1}^{V} r_i^* \end{cases}.$$

Algorithm 2 requires solving S from $\mathcal{L}$ with an additional linear equation $\mathcal{E}$. We denote it by $\mathrm{solve}(\mathcal{L},\mathcal{E})$.

Algorithm 2 Optimal Solution When $J < V-K$

```
Input: $S, \mu_1, \mu_2, \ldots, \mu_V, V, K, J, Q$
Output: $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_\Pi$, $\boldsymbol{l}^*, \boldsymbol{r}^*, T^*$
1: Initialize $\mathcal{N}, \mathcal{N}_0^{(0)}, \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_\Pi$ all to $\varnothing$;
2: $\mathcal{S}_0 := \{S_0^{(0)}\}$, $S_0^{(0)} := S_0$;
3: for $k := k_{start}, k_{start}+1, \ldots, \Pi-1$ do
4:     $S_{k+1}^{(0)} := \mathrm{solve}(\mathcal{L}(\mathcal{N}, Q-q_k), l_1^* = \mu_{(k+1)}T^*)$;
5:     for $z := 1, 2, \ldots, |\mathcal{N}|$ do
6:         $n := \min(\mathcal{N})$;
7:         $S_{k+1}^{(z)} := \mathrm{solve}(\mathcal{L}(\mathcal{N}, Q-q_k), l_V^* = \mu_{(n)}T^*)$;
8:         if $\max\{\cup_{i=0}^{k+1}\mathcal{S}_i\} < S_{k+1}^{(z)} < S_{k+1}^{(0)}$ then
9:             $\mathcal{S}_{k+1} := \mathcal{S}_{k+1} \cup \{S_{k+1}^{(z)}\}$;
10:            $\mathcal{N} := \mathcal{N} \setminus \{n\}$;
11:            $\mathcal{N}_{k+1}^{(z)} := \mathcal{N}$;
12:            $S_{k+1}^{(0)} := \mathrm{solve}(\mathcal{L}(\mathcal{N}, Q-q_k), l_1^* = \mu_{(k+1)}T^*)$;
13:        else
14:            break;
15:        end if
16:    end for
17:    $\mathcal{N} := \mathcal{N} \cup \{k+1\}$;
18:    $\mathcal{N}_{k+1}^{(0)} := \mathcal{N}$;
19:    $\mathcal{S}_{k+1} := \mathcal{S}_{k+1} \cup \{S_{k+1}^{(0)}\}$;
20: end for
21: if $S_i^{(i')} < S \le S_j^{(j')}$, where $\{S_r^{(r')} \in \cup_{i=0}^{\Pi}\mathcal{S}_i : S_i^{(i')} < S_r^{(r')} < S_j^{(j')}\} = \varnothing$ then
22:     $\boldsymbol{l}^*$, $\boldsymbol{r}^*$ and $T^*$ are obtained by solving $\mathcal{L}(\mathcal{N}_i^{(i')}, x(S))$ with the given $S$;
23: else if $S > S_\Pi^{(0)}$ then
24:     $\boldsymbol{l}^*$, $\boldsymbol{r}^*$ and $T^*$ are the same as the case where $S = S_\Pi^{(0)}$.
25: end if
```

D. Majorization Property

In this subsection, we compare the performance of two multi-cloud systems with the same total computing power. An interesting result is that the more uneven the computing rates of the clouds are, the longer the optimal computation time will be. To capture precisely the degree of "unevenness" of cloud computing rates, we adopt a mathematical notion from [45] called *majorization* in Definition 1. We can see "$\boldsymbol{x} \succ \boldsymbol{y}$" describes the intuition that the components of $\boldsymbol{x}$ are "more spread out" or "less nearly equal" than those of $\boldsymbol{y}$. A related concept called *Schur-concavity* is also defined below [45]:

Definition 10: A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be Schur-concave if $\boldsymbol{x} \succ \boldsymbol{y}$ implies $f(\boldsymbol{x}) \le f(\boldsymbol{y})$. It is said to be Schur-convex if $\boldsymbol{x} \succ \boldsymbol{y}$ implies $f(\boldsymbol{x}) \ge f(\boldsymbol{y})$.

The following lemma is an important conclusion about Schur-concave functions, whose proof can be found in [45, p.92]:

Lemma 19 ([45]): If $I \subset \mathbb{R}$ is an interval and $f : I \to \mathbb{R}$ is concave, then

$$\Phi(x) = \sum_{i=1}^{n} f(x_i)$$

is Schur-concave on I^n. Consequently, $\boldsymbol{x} \succ \boldsymbol{y}$ on I^n implies $\Phi(\boldsymbol{x}) \le \Phi(\boldsymbol{y})$.

In a (V, K, J) system, we assume the computing rate vector of the V clouds, denoted as $\boldsymbol{\mu} \triangleq (\mu_1, \dots, \mu_V)$, is indexed in a way such that its components are sorted in decreasing order. Consider two (V, K, J) systems with computing rate vectors $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$, where $\boldsymbol{\mu}^{(1)} \succ \boldsymbol{\mu}^{(2)}$. We have the following result on their optimal computing times:

Proposition 20 (Proof of Theorem 3): Let $f(\boldsymbol{\mu})$, where $f : \mathbb{R}^V \to \mathbb{R}$, be defined as the optimal computation time T^* for a system with computing rate vector μ. Given any $S \geq S_0$, f is Schur-convex.

Proof: Define $\mathcal{L}(S) \triangleq \{\boldsymbol{l} : \|\boldsymbol{l}\|_1 \leq S,\ l_1 \geq \cdots \geq l_V,\ \sum_{i=V-K+1}^{V} l_i - \sum_{i=1}^{J} l_i = 1\}$ as the set of storage allocation vectors which satisfy (33), (34), and (35). Given any $\boldsymbol{l} \in \mathcal{L}(S)$, the optimal computing time can be obtained by solving:

$$\min_{\boldsymbol{r}} T \tag{62}$$

subject to

$$\|\boldsymbol{r}\|_1 = \sum_{i=V-K+1}^{V} l_i \tag{63}$$

$$0 \leq r_i \leq \min\{\mu_i T, l_i\},\ \forall i \in \mathcal{V} \tag{64}$$

It is easy to see that at the optimal solution $\boldsymbol{r}^*$, equality must hold in (64), for otherwise there exist some clouds completing computation tasks in advance $(r_i < \mu_i T)$ without using all the local data $(r_i < l_i)$, which means those clouds could compute more to reduce the workload of bottleneck clouds so that a lower value of T can be achieved by adjusting the values of r_i's while keeping their sum constant as in (63). Hence, the function $f(\boldsymbol{\mu})$ can then be represented as

$$f(\boldsymbol{\mu}) = \min_{\boldsymbol{l} \in \mathcal{L}(S)} \varphi_{\boldsymbol{l}}(\boldsymbol{\mu}),$$

where

$$\varphi_{\boldsymbol{l}}(\boldsymbol{\mu}) \triangleq \min \Big\{ T \in \mathbb{R} : g_{\boldsymbol{l}}(\boldsymbol{\mu}, T) = \sum_{i=V-K+1}^{V} l_i \Big\}, \tag{65}$$

and

$$g_{\boldsymbol{l}}(\boldsymbol{\mu}, T) \triangleq \sum_{i=1}^{V} \min\{\mu_i T, l_i\}.$$

First, we show that $\varphi_{\boldsymbol{l}}(\boldsymbol{\mu})$ is Schur-convex for any $\boldsymbol{l} \in \mathcal{L}(S)$. Consider any two systems with the same storage allocation vector $\boldsymbol{l}$ and computing rate vectors $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$, respectively, such that $\boldsymbol{\mu}^{(1)} \succ \boldsymbol{\mu}^{(2)}$. Since $\min\{\mu_i T, l_i\}$ is a concave function of μ_i, by Lemma 19, $g_{\boldsymbol{l}}(\boldsymbol{\mu}, T)$ is a Schur-concave function of $\boldsymbol{\mu}$, i.e., $g(\boldsymbol{\mu}^{(1)}, T) \leq g(\boldsymbol{\mu}^{(2)}, T)$. Define $T^*_{\boldsymbol{l},i} \triangleq \varphi_{\boldsymbol{l}}(\boldsymbol{\mu}^{(i)})$ for $i = 1, 2$. By the definition of $\varphi_{\boldsymbol{l}}(\boldsymbol{\mu})$, we have $g(\boldsymbol{\mu}^{(1)}, T^*_{\boldsymbol{l},1}) = g(\boldsymbol{\mu}^{(2)}, T^*_{\boldsymbol{l},2})$, since both of them equal $\sum_{i=V-K+1}^{V} l_i$. Combining the two results, we have

$$g(\boldsymbol{\mu}^{(2)}, T^*_{\boldsymbol{l},2}) \leq g(\boldsymbol{\mu}^{(2)}, T^*_{\boldsymbol{l},1}).$$

As $g(\boldsymbol{\mu}, T)$ is an increasing function of T, we conclude

$$\varphi_{\boldsymbol{l}}(\boldsymbol{\mu}^{(1)}) \triangleq T^*_{\boldsymbol{l},1} \geq T^*_{\boldsymbol{l},2} \triangleq \varphi_{\boldsymbol{l}}(\boldsymbol{\mu}^{(2)}).$$

Next, we show that $f(\boldsymbol{\mu})$ is Schur-convex. For $i = 1, 2$, let $\boldsymbol{l}^*_i \in \mathcal{L}(S)$ be an optimal vector such that $\varphi_{\boldsymbol{l}}(\boldsymbol{\mu}^{(i)})$ is minimal. We then have

$$f(\boldsymbol{\mu}^{(1)}) = \varphi_{\boldsymbol{l}^*_1}(\boldsymbol{\mu}^{(1)}) \geq \varphi_{\boldsymbol{l}^*_1}(\boldsymbol{\mu}^{(2)}) \geq \varphi_{\boldsymbol{l}^*_2}(\boldsymbol{\mu}^{(2)}) = f(\boldsymbol{\mu}^{(2)}),$$

where the first inequality follows from the Schur-convexity of $\varphi_{\boldsymbol{l}}(\boldsymbol{\mu})$ and the second inequality follows from the optimality of $\boldsymbol{l}^*_2$ for the system with $\boldsymbol{\mu}^{(2)}$. $\square$

Proposition 20 proves Theorem 3, which implies that heterogeneity negatively impacts the tradeoff between storage and computation. If the total computing power is fixed, homogeneous multi-cloud systems provide faster computing service than heterogeneous systems under the same storage budget. In general, the more even the computing rates are, the shorter the computing time a system can achieve.

VII. Numerical Analysis and Experiments Using Amazon EC2 Instances

In this section, we implement matrix-vector multiplication $\mathbf{Ax}$ over a finite field $\mathbb{F}_q$ reliably and securely in a (V, K, J) multi-cloud system. For practical scenario where input matrix elements are real numbers, our proposed coding scheme can also be naturally applied. For example, a real number can be quantized and represented by a fixed-point number, which is essentially the same as an integer. The matrix and vector entries, regarded as integers, can then be embedded into a finite field of size greater than the possible range of an output vector entry.

The UT code is proved capacity achieving, and will be used throughout this section. We will demonstrate the performance of our optimal resource allocation scheme numerically based on our analytical results and test it experimentally on Amazon EC2 clusters in the following subsections.

A. Benchmark Schemes

To evaluate the performance of our optimized storage allocation, two simple allocation schemes, namely, equal allocation and proportional allocation, will be used as benchmarks for comparison. Given a storage budget S, to apply UT codes, both the equal allocation vector $\boldsymbol{l} = \frac{S}{V}\mathbf{1}_V$ and the proportional allocation vector $\boldsymbol{l} = \frac{S}{\|\boldsymbol{\mu}\|_1}\boldsymbol{\mu}$ must fulfill $\omega \leq \varphi$. The former is equivalent to $S \geq S_0$, and the latter is equivalent to $S \geq S_{\text{pro}}$, where

$$S_{\text{pro}} \triangleq \frac{\|\boldsymbol{\mu}\|_1}{\min\limits_{\mathcal{K} \subset_K \mathcal{V}} \sum\limits_{i \in \mathcal{K}} \mu_i - \max\limits_{\mathcal{J} \subset_J \mathcal{V}} \sum\limits_{i \in \mathcal{J}} \mu_i} \tag{66}$$

is positive and bounded, denoting the minimum storage budget for proportional allocation. It should be noted that proportional allocation may not be feasible in our system. Moreover, unlike our optimized allocation, an increase in storage budget for equal allocation and proportional allocation cannot reduce the computation time. For equal allocation with $\boldsymbol{l} = \frac{S}{V}\mathbf{1}_V$, the corresponding optimal computing time T^*_{equal}, obtained by solving problem (27), is

$$T^*_{\text{equal}} = \frac{S(K - Q)}{V \sum_{i \in \mathcal{B}_0} \mu_i},$$

which linearly increases with the growth of S. The proof is the same as Lemma 14, except for replacing l_0 with S/V. In a similar manner, we can show that the optimal computing time of proportional allocation also linearly increases as S increases. Therefore, equal allocation and proportional allocation do not need to fully utilize the storage budget, but should just store S_0 and S_{pro}, respectively. The settings of the two schemes are summarized below:

(1) **Equal allocation with UT code**: The confidential matrix is encoded by UT code. With the fixed storage allocation vector $\boldsymbol{l} = \boldsymbol{l}_0$, the load vector $\boldsymbol{r} = \boldsymbol{r}_0$ is optimal in minimizing the average computing time in problem (27).

(2) **Proportional allocation with UT code**: The confidential matrix is encoded by UT code. With the fixed storage allocation vector $\boldsymbol{l} = \frac{S_{\text{pro}}}{\|\boldsymbol{\mu}\|_1}\boldsymbol{\mu}$, the load vector $\boldsymbol{r}$ is optimized to minimize the average computing time by solving problem (27).

B. Numerical Results

Consider a $(V, K, J) = (6, 5, 2)$ system, which requires a minimum storage budget $S_0 = 2m$. The computing rate vector of the six clouds, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_6)$, determines the value of Q, which is the number of non-bottleneck clouds under the minimum storage budget. Given a storage budget $S \geq S_0$ and a data matrix of dimension $m \times s$, the following four scenarios of various computing rates are considered:

- **Scenario 1**: $\boldsymbol{\mu} = \boldsymbol{\mu}^{(1)} \triangleq (9, 7, 3, 3, 2, 1)$ and $Q = 2$.
- **Scenario 2**: $\boldsymbol{\mu} = \boldsymbol{\mu}^{(2)} \triangleq (8, 6, 5, 3, 2, 1)$ and $Q = 3$.
- **Scenario 3**: $\boldsymbol{\mu} = \boldsymbol{\mu}^{(3)} \triangleq (7, 6, 5, 4, 2, 1)$ and $Q = 4$.
- **Scenario 4**: $\boldsymbol{\mu} = \boldsymbol{\mu}^{(4)} \triangleq (5, 4, 4, 4, 4, 4)$ and $Q = 0$.

Note that the system in each scenario has the same total computational rate of $\|\boldsymbol{\mu}\|_1 = 25$. Moreover, it is easy to check that $\boldsymbol{\mu}^{(1)} \succ \boldsymbol{\mu}^{(2)} \succ \boldsymbol{\mu}^{(3)} \succ \boldsymbol{\mu}^{(4)}$. Same as section VI, the proposed storage allocation vector $\boldsymbol{l}$ and load vector $\boldsymbol{r}$ are normalized by m, i.e., we set $m = 1$ in the simulations.

We first characterize the tradeoff between the optimal computing time T^* and storage budget S for the four different scenarios, which is illustrated in Fig. 4. The optimal computing time is shown to be a piecewise linear non-increasing function of the storage budget. For Scenarios 1 and 4, in which $Q \leq J$, increase in storage budget cannot reduce computing time. For Scenarios 2 and 3, in which $Q > J$, the optimal computing time decreases with the growth of storage budget, and its decreasing rate also declines due to the number of non-bottleneck clouds getting smaller in (61). When the storage budget exceeds a certain threshold, the optimal computation time levels off and cannot be reduced by further increasing the storage budget. The results also verify that there are $\Pi = Q - J$ turning points. Furthermore, it can be seen from Fig. 4 that $T_1^* \geq T_2^* \geq T_3^* \geq T_4^*$ for any feasible storage budget, which verifies Theorem 3 and demonstrates that uneven computing rates of clouds lead to longer computing time.

Next, we compare our proposed scheme with the two benchmarks described in the previous subsection. Since proportional allocation is infeasible in Scenario 1, we only consider Scenarios 2, 3, and 4. Fig. 5 shows that our proposed allocation

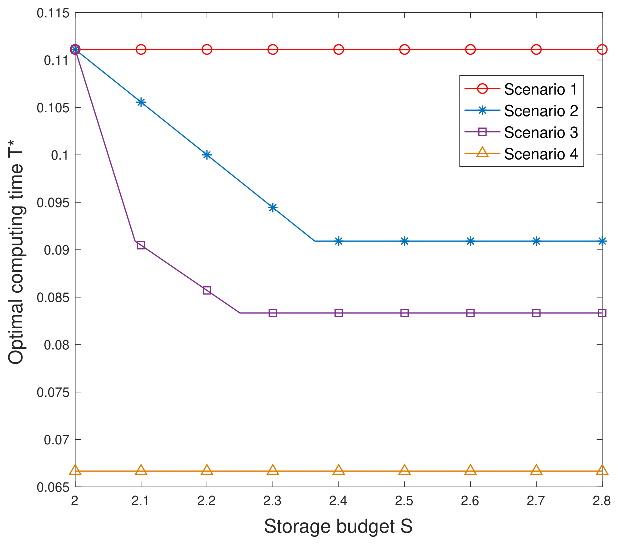

Fig. 4. Tradeoff between storage budget and optimal computing time in four scenarios.

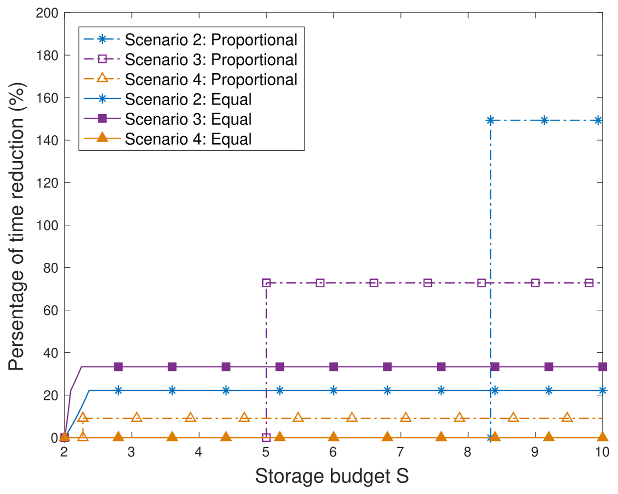

Fig. 5. Time reduction by our proposed allocation compared with equal allocation and proportional allocation in three scenarios.

outperforms proportional allocation, plotted in broken lines, in all three scenarios. This is because the total computational workload of proportional allocation is much higher, outweighing the benefit of having all clouds compute in parallel from start to finish. It assigns more storage amount to the fastest J clouds, but to ensure data security in these J clouds, UT code adds a large number of random keys, requiring much more calculations to finish the computing task. That explains why the minimum storage budget of proportional allocation, S_{pro}, is larger than the minimum feasible budget, S_0. Equal allocation is the same as our proposed allocation with storage budget fixed at S_0. Its performance curves are shown by solid lines in Fig. 5. For Scenarios 2 and 3, in which $Q > J$, our optimized allocation can take advantage of the available storage budget to reduce the computing time until a threshold is reached. Although equal allocation has less computational workload compared with ours, it does not make a good use of the cloud computing power as the faster clouds could do more tasks to minimize the computing time. For Scenario 4, in which $Q < J$, the performance of equal allocation is

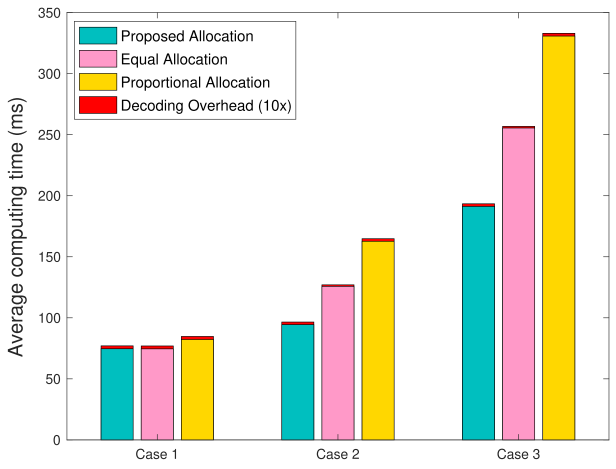

Fig. 6. Performance comparison of three schemes. Decoding time is expanded tenfold for easy visualization.

identical to ours, since the computing time cannot decline under any storage budget. In summary, our allocation scheme outperforms both benchmarks in most cases as it combines the benefit of equal allocation (less computational workload) and proportional allocation (assigning more data to faster clouds). In Scenarios 2 and 3, as the storage budget increases, our optimal allocation scheme can provide gains in the computing time of up to $20\% - 35\%$ over equal allocation, and up to $70\% - 150\%$ over proportional allocation.

C. Experiments Using Amazon EC2 Instances

We use MPICH2 [46] to implement our proposed scheme and conduct experiments on Amazon EC2 clusters. A cluster, consisting of a master and a group of workers of instance type `t2.micro`, is used to emulate a cloud. Several clusters are used to emulate multiple clouds. Cloud i contains μ_i worker instances, for $i = 1, 2, \ldots, V$. Although μ_i may not equal exactly the actual computing rate of Cloud i, it does reflect the computing power of the cloud. In our experiments, we consider the same $(V, K, J) = (6, 5, 2)$ system as in the previous subsection.

In the first experiment, we set the storage budget to $S = S_{\text{pro}} * m$, where $m = 15840$. The optimal allocation $\boldsymbol{l}^*$ and $\boldsymbol{r}^*$ can then be obtained according to the computing rate vector $\boldsymbol{\mu}$. For example, given $\boldsymbol{\mu} = (7, 6, 5, 4, 2, 1)$, we can obtain $\boldsymbol{l}^* = (5, 5, 5, 5, 4, 3) * \frac{1}{12}$ and $\boldsymbol{r}^* = (5, 5, 5, 4, 2, 1) * \frac{1}{12}$. Choosing $\hat{m} = m/b = 12$, we apply a $(\hat{\boldsymbol{l}}, \hat{\boldsymbol{r}}, \hat{m})$ UT code to encode the confidential matrix and perform secure multi-cloud computing, where $\hat{\boldsymbol{l}} = \hat{m}\boldsymbol{l}^*$ and $\hat{\boldsymbol{r}} = \hat{m}\boldsymbol{r}^*$. In a similar manner, equal allocation with UT code and proportional allocation with UT code can also be implemented. For performance comparison of the three schemes, the following three cases are considered:

- **Case 1**: $s = 5000$ and $\boldsymbol{\mu} = (5, 4, 4, 4, 4, 4)$.
- **Case 2**: $s = 5000$ and $\boldsymbol{\mu} = (7, 6, 5, 4, 2, 1)$.
- **Case 3**: $s = 10000$ and $\boldsymbol{\mu} = (7, 6, 5, 4, 2, 1)$.

Fig. 6 shows the computing time of the three schemes for the above three cases. To better assess the performance of each scheme, the time overhead due to decoding is also measured. It turns out the decoding overhead is much smaller or even negligible compared with the computing time, so for easy visualization, we magnify the decoding time tenfold in the figure. For Case 1 in which only one cloud is slightly faster than the others, our proposed scheme performs almost the same as equal allocation. For Cases 2 and 3 in which the cloud computing speeds are more diverse, our scheme achieves 32.99% and 33.52% time reduction, respectively. For all three cases, proportional allocation does not perform well. Our scheme can reduce its computing time by 10.14%, 72.10%, and 72.91%, respectively. These results are roughly consistent with the simulation results in Fig. 5. We can see increasing the value of s does not affect the performance gain of our proposed allocation scheme compared to the two benchmarks.

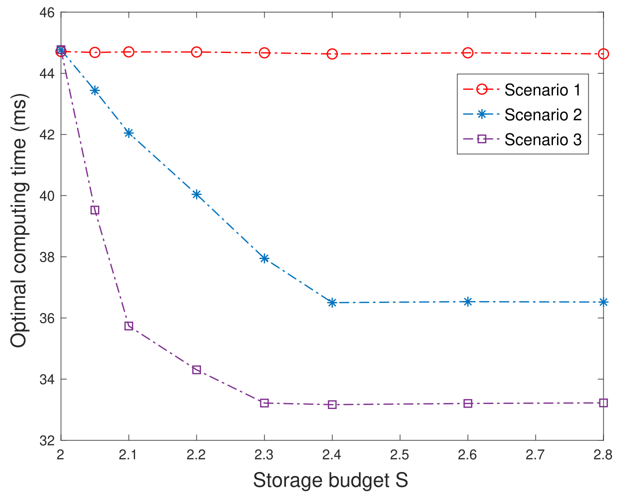

Fig. 7. Tradeoff between storage budget and optimal computing time in three scenarios.

In the second experiment, we investigate the tradeoff between between the optimal computing time and storage budget for three different cloud configurations (Scenarios 1, 2 and 3 in Fig. 4). We set $s = 1000$ and vary the storage budget S along the discrete points $(2.0, 2.05, 2.1, 2.2, 2.3, 2.4, 2.6, 2.8) * m$, where $m = 27720$. The experimental results in Fig. 7 are consistent with the numerical results in Fig. 4, which corroborates our previous theoretical analysis. Furthermore, $T_1^* \geq T_2^* \geq T_3^*$ for any feasible storage budget due to the majorization property, verifying that uneven computing rates of clouds leads to longer computing time.

VIII. Conclusion

Coded distributed computing with perfect secrecy and heterogeneity is studied. The secrecy capacity, defined as the maximum achievable code rate, is obtained explicitly, and is shown to be achievable if and only if the encoded data is equally allocated to all clouds. While a major goal of distributed computing is to shorten the computation time, equal allocation, though optimal in terms of storage efficiency, may not be the best choice for a heterogeneous system. A fundamental tradeoff between storage budget and computation time is characterized in the information-theoretic sense, which can be achieved with linear decoding complexity by the UT code. The storage budget constrains the code rate to be above a certain threshold. Given a feasible storage budget, the computation time is usually minimized with unequal allocation, at the cost of a lower code rate than secrecy capacity. Moreover, the tradeoff curve shifts towards the origin if the computing rates become more even. The implication is that heterogeneity lengthens computing time. Such a phenomenon was not discussed before. It is hoped that this study sheds light on efficient design of multi-cloud systems in particular and coded distributed computing systems in general. Future works may consider generalizing the results to a broader setting, e.g., two-sided secure matrix multiplication, in which both matrices are required to be secure, and batch matrix multiplication, in which a collection of matrix products need to be computed.

APPENDIX A
PROOF OF PROPOSITION 9

Proof: Let $\mathbf{G}_0^*$ be an arbitrary $\omega \times \omega$ submatrix of $\mathbf{G}_0$. The proof of $\mathbf{G}_0^*$ being full rank is based on the fact that adding a scalar multiple of one column to another column does not change the value of its determinant, and interchanging any pair of columns or rows of the matrix multiplies its determinant by -1.

Rewrite $\mathbf{G}_0$ as

$$\mathbf{G}_0 \triangleq \begin{bmatrix}\mathbf{G}_{01} & \mathbf{G}_{02} & \mathbf{G}_{03}\end{bmatrix} \triangleq \begin{bmatrix}\mathbf{I}_\theta & \mathbf{U}_{12} & \mathbf{U}_{13} \\ \mathbf{0} & \mathbf{I}_{\hat{m}} & \mathbf{U}_{23}\end{bmatrix}.$$

Assume we pick p $(p \le \theta)$ columns from $\mathbf{G}_{01}$, q $(q \le \hat{m})$ columns from $\mathbf{G}_{02}$, and $\omega - p - q$ columns from $\mathbf{G}_{03}$ to construct $\mathbf{G}_0^*$, then after adding a scalar multiple of one column to another column and interchanging some rows and columns, the determinant of $\mathbf{G}_0^*$ satisfies

$$|\det(\mathbf{G}_0^*)| = |\det \begin{bmatrix}\mathbf{I}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{\omega-p}\end{bmatrix}| = |\det(\mathbf{M}_{\omega-p})|,$$

where $\mathbf{I}_p$ and $\mathbf{M}_{\omega-p}$ are square matrices with size p and $\omega-p$ respectively.

$$\mathbf{M}_{\omega-p} = \begin{bmatrix}\mathbf{U}'_{11} & \mathbf{U}'_{12} & \mathbf{U}'_{13} \\ \mathbf{I}_q & \mathbf{0} & \mathbf{U}'_{23}\end{bmatrix} = \begin{bmatrix}\mathbf{U}'_{11} & \mathbf{D} \\ \mathbf{I}_q & \mathbf{B}\end{bmatrix},$$

where $\mathbf{U}'_{ij}$'s are random matrices with independently and uniformly distributed entries, and $\mathbf{I}_q$ and $\mathbf{D}$ are square matrices with size q and $\omega - p - q$ respectively. Note that the sizes of the two row groups are $\omega - p - q$ and q, respectively, while those of the three column groups are q, $\hat{m} - q$ and $\theta - p$, respectively.

By Schur's determinant identity [43, p.475], we have

$$\begin{aligned}|\det(\mathbf{M}_{\hat{m}+\theta-p})| &= |\det(\mathbf{I}_q)||\det(\mathbf{D} - \mathbf{U}'_{11}\mathbf{I}_q^{-1}\mathbf{B})| \\ &= q * |\det(\mathbf{D} - \mathbf{U}'_{11}\mathbf{B})|,\end{aligned}$$

where $\mathbf{D} - \mathbf{U}'_{11}\mathbf{B}$ is a random matrix, whose entries are all independent and uniformly distributed in $\mathbb{F}_q$. When the field size q goes to infinity, the probability of $\mathbf{D} - \mathbf{U}'_{11}\mathbf{B}$ being of full rank (i.e., $det(\mathbf{G}_0^*) \neq 0$) approaches 1, which implies that $\mathbf{G}_0$ has property (1) with probability 1.

In a similar way, we can prove that $\mathbf{G}_0$ also has property (2) with probability 1. □

APPENDIX B
PROOF OF LEMMA 11

Proof: The reverse direction is obvious. If $S \ge \frac{V}{K-J}$, it is easy to check that $\boldsymbol{l} = \frac{1}{K-J}\mathbf{1}_V$ and $\boldsymbol{r} = \frac{K}{V(K-J)}\mathbf{1}_V$ satisfy all the constraints and together form a feasible solution.

Next, assume the problem is feasible. Let the allocation vector $\boldsymbol{l} \triangleq (l_1, l_2, \ldots, l_V)$ and load vector $\boldsymbol{r} \triangleq (r_1, r_2, \ldots, r_V)$ be a feasible solution. We sort $l_1, l_2, \ldots, l_V$ in decreasing order and denote them by $l_{(1)}, l_{(2)}, \ldots, l_{(V)}$ such that

$$l_{(1)} \ge l_{(2)} \ge \cdots \ge l_{(V)} \ge 0. \tag{67}$$

Rewrite (29) as

$$\begin{aligned}1 &\le \sum_{i=V-K+1}^{V} l_{(i)} - \sum_{i=1}^{J} l_{(i)} \\ &= \sum_{i=J+1}^{V} l_{(i)} - \sum_{i=1}^{V-K} l_{(i)} \\ &= \sum_{i=V-K+J+1}^{V} l_{(i)} - \sum_{i=1}^{V-K} [l_{(i)} - l_{(J+i)}] \\ &\le \sum_{i=V-K+J+1}^{V} l_{(i)},\end{aligned} \tag{68}$$

with equality if and only if $l_{(1)} = l_{(2)} = \cdots = l_{(J+V-K)}$. Since the sequence is in decreasing order, (68) implies

$$l_{(V-K+J+1)} \ge \frac{1}{K-J}, \tag{69}$$

and that the equality holds if and only if $l_{(V-K+J+1)} = l_{(V-K+J+2)} = \cdots = l_{(V)} = \frac{1}{K-J}$. From (28),

$$\begin{aligned}S &\ge \sum_{i=1}^{V-K+J} l_{(i)} + \sum_{i=V-K+J+1}^{V} l_{(i)} \\ &\overset{(a)}{\ge} \frac{V-K+J}{K-J} + 1 \\ &= \frac{V}{K-J},\end{aligned} \tag{70}$$

where (a) follows from (68) and (69). The condition in (70) is thus necessary, which proves the forward direction. In particular, if $S = \frac{V}{K-J}$, (a) must hold with equality, which implies $l_{(1)} = l_{(2)} = \cdots = l_{(V)} = \frac{1}{K-J}$. □

APPENDIX C
PROOF OF LEMMA 12

Proof: Given a feasible instance, it is easy to see that the feasible region is closed and bounded. Since $\max_{i \in \mathcal{V}} r_i/\mu_i$ is a continuous function of $\boldsymbol{r}$, by Weierstrass Theorem [44, p.90], an optimal solution exists. Let $\boldsymbol{l}'$ be an optimal allocation vector, and $\boldsymbol{r}'$ be the corresponding load vector. The resultant computation time is denoted by $T^* \triangleq \max_{i \in \mathcal{V}} \frac{r'_i}{\mu_i}$.

Suppose $l'_i < l'_j$, where $i < j$. We distinguish between two cases. In the first case, $r'_i \ge r'_j$. We then swap the values of l'_i and l'_j, which would still be feasible without affecting the value of T^*. In the second case, $r'_i < r'_j$. We swap the values of (l'_i, r'_i) and (l'_j, r'_j), which would still be feasible. The computation time would not be increased, since $\mu_i \geq \mu_j$. (This could not reduce the computation time, since T^* is assumed to be optimal.) By repeating the above procedure, we obtain optimal vectors $\boldsymbol{r}^*$ and $\boldsymbol{l}^*$, where $l_1^* \geq l_2^* \geq \cdots \geq l_V^*$.

If (29) is active at $\boldsymbol{l}^*$, we are done. Otherwise, we rewrite (29) and (31), respectively, as

$$\sum_{i=V-K+1}^{V} l_i^* - \sum_{i=1}^{J} l_i^* = \sum_{i=J+1}^{V} l_i^* - \sum_{i=1}^{V-K} l_i^* \geq 1, \tag{71}$$

$$\sum_{i\in\mathcal{V}} r_i^* = \sum_{i=V-K+1}^{V} l_i^*. \tag{72}$$

We are going to adjust $\boldsymbol{l}^*$ and $\boldsymbol{r}^*$ to produce another optimal solution at which equality in (71) holds. Define $\mathcal{I} \triangleq \{Z+1, Z+2, \ldots, V\}$, where $Z \triangleq \max\{J, V-K\}$. First, notice that there must exist $\delta_i \geq 0$ for all $i \in \mathcal{I}$, such that reducing l_i^* by δ_i for all $i \in \mathcal{I}$ could meet the lower bound in (71). Next, to ensure that (30) still holds, we can reduce r_i^* to $(r_i^* - \delta_i)^+$ for all $i \in \mathcal{I}$. If $r_i^* < \delta_i$ for some $i \in \mathcal{I}$, the left hand side of (72) will be strictly greater than the right hand side. If that is the case, we reduce the values of those r_i^*'s that are positive such that equality holds. This can always be done, since r_i's are free to decrease to any non-negative values. The above change clearly will not increase the value of T^*. □

Appendix D
Proof of Lemma 17

Proof: To establish $T_Q(\Delta s)$ as a lower bound on T, following (47), we want to find an upper bound on $\sum_{i=J+1}^{Q} \Delta l_i$, which depends only on Δs.

First, consider the case where $Q > \max\{J, V-K\}$ with λ defined as the first expression in (50). From (44) and (46) we have

$$\sum_{i\in\mathcal{V}} \Delta l_i = \sum_{i=1}^{V-K} \Delta l_i + \sum_{i=1}^{J} \Delta l_i \leq \Delta s. \tag{73}$$

Define $\sigma \triangleq \sum_{i=1}^{V-K} \Delta l_i + \sum_{i=1}^{J} \Delta l_i$. Then $\lambda\sigma$ is the mean of the $1/\lambda$ terms in the two summations.

If $Q > J \geq V-K$, by (45), Δl_J is the smallest term in the two summations of (73). Therefore, $\Delta l_J \leq \lambda\sigma \leq \lambda\Delta s$, implying

$$\sum_{i=J+1}^{Q} \Delta l_i \leq (Q-J)\Delta l_J \leq (Q-J)\lambda\Delta s.$$

If $Q > V-K > J$. By the same argument, Δl_{V-K} is the smallest term in the two summations of (73). Therefore, $\Delta l_{V-K} \leq \lambda\Delta s$, implying

$$\begin{aligned}\sum_{i=J+1}^{Q} \Delta l_i &= \sum_{i=J+1}^{V-K} \Delta l_i + \sum_{i=V-K+1}^{Q} \Delta l_i \\ &\leq \sum_{i=J+1}^{V-K} \Delta l_i + (Q-V+K)\lambda\Delta s \\ &= \sigma - 2\sum_{i=1}^{J} \Delta l_i + (Q-V+K)\lambda\Delta s \\ &\overset{(a)}{\leq} \sigma - 2J\lambda\sigma + (Q-V+K)\lambda\Delta s \\ &= (V-K-J)\lambda\sigma + (Q-V+K)\lambda\Delta s \\ &\overset{(b)}{\leq} (Q-J)\lambda\Delta s. \end{aligned} \tag{74}$$

where (a) follows from the fact that the sum of the greatest J terms in the two summations of (73) must be no less than J times the mean of all $1/\lambda$ terms, and (b) comes from $\sigma \leq \Delta s$.

Hence, for $Q > \max\{J, V-K\}$, we obtain

$$\sum_{i=J+1}^{Q} \Delta l_i \leq (Q-J)\lambda\Delta s, \tag{75}$$

which implies $T_Q(\Delta s)$ is a lower bound on T. Recall that equality holds in (47) if and only if

$$\mathcal{B}_{\Delta s} \supseteq \mathcal{B}_0 \text{ and } r_i^* = l_i^* \ \forall i \in \bar{\mathcal{B}}_0. \tag{76}$$

It is straightforward to check that $T_Q(\Delta s)$ can be achieved by a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ if and only if it satisfies $\Delta l_1 = \cdots = \Delta l_Q = \lambda\Delta s$ and (76).

Next, consider the case where $V-K \geq Q > J$ with λ defined as the second expression in (50). Define

$$\delta' \triangleq \sum_{i=1}^{Q} \Delta l_i = \sum_{i=1}^{J} \Delta l_i + \sum_{i=J+1}^{Q} \Delta l_i. \tag{77}$$

Because the sum of the greatest J terms in the two summations of (77) must be no less than J times the mean of all Q terms, we have

$$\sum_{i=1}^{J} \Delta l_i \geq \frac{J}{Q}\delta'. \tag{78}$$

Then,

$$\sum_{i=J+1}^{Q} \Delta l_i = \delta' - \sum_{i=1}^{J} \Delta l_i \leq \frac{(Q-J)}{Q}\delta'. \tag{79}$$

From (44), (45) and (46), we have

$$\begin{aligned}\Delta s &\geq \delta' + \sum_{i=Q+1}^{V-K} \Delta l_i + \sum_{i=V-K+1}^{V} \Delta l_i \\ &\geq \delta' + (V-K-Q)\Delta l_{V-K+1} + \sum_{i=V-K+1}^{V} \Delta l_i \\ &\overset{(c)}{\geq} \delta' + (V-K-Q)\frac{\sum_{i=V-K+1}^{V} \Delta l_i}{K} + \sum_{i=V-K+1}^{V} \Delta l_i \\ &= \delta' + \frac{V-Q}{K}\sum_{i=1}^{J} \Delta l_i \\ &\overset{(d)}{\geq} \frac{KQ + J(V-Q)}{KQ}\delta', \end{aligned} \tag{80}$$

where (c) follows from the fact that the largest term of a summation must be no less than its mean and (d) comes from (78).

Combining (79) and (80), we finally obtain

$$\sum_{i=J+1}^{Q} \leq (Q-J)\lambda\Delta s. \tag{81}$$

Therefore, $T_Q(\Delta s)$ is a lower bound on T according to (47). It is straightforward to check that $T_Q(\Delta s)$ can be achieved by a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ if and only if it satisfies $\Delta l_1 = \cdots = \Delta l_Q = \lambda\Delta s$, $\Delta l_{Q+1} = \cdots = \Delta l_V = \frac{J}{K}\lambda\Delta s$ and (76). □

Appendix E
Proof of Proposition 18

Proof: Following the statement of the proposition, we divide the proof into two cases. Furthermore, the second case is further divided into two sub-cases.

Case 1: $Q > J \geq V - K$.

Lemma 17 shows that $T_Q(\Delta s)$ can be achieved if and only if there exists a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ such that $r_i^* = \mu_i T_Q(\Delta s)$ $\forall i \in \mathcal{B}_0$ and $r_i^* = l_i^* = l_0 + \lambda\Delta s$ $\forall i \in \bar{\mathcal{B}}_0$, which is equivalent to

$$\frac{r_1^*}{\mu_1} = \frac{l_1^*}{\mu_1} = \frac{l_0+\lambda\Delta s}{\mu_1} \leq \cdots \leq \frac{r_Q^*}{\mu_Q} = \frac{l_Q^*}{\mu_Q} = \frac{l_0+\lambda\Delta s}{\mu_Q} \leq$$
$$T_Q(\Delta s) = \frac{r_{Q+1}^*}{\mu_{Q+1}} = \cdots = \frac{r_V^*}{\mu_V}. \tag{82}$$

When $\Delta s = 0$, Lemma 14 shows that the above condition is true and $T_Q(0)$ can be achieved. As Δs grows from 0, $r_i^* = l_i^* = l_0 + \lambda\Delta s$ strictly increases from l_0 and grows without bound for all $i \in \bar{\mathcal{B}}_0$, while $T_Q(\Delta s)$ strictly decreases and $\frac{r_i^*}{\mu_i}$ is always equal to $T_Q(\Delta s)$ for all $i \in \mathcal{B}_0$. Therefore, there must exist $\Delta s_1 \triangleq S_1^{(0)} - S_0 > 0$ such that

$$\frac{r_i^*}{\mu_i} = \frac{l_0+\lambda\Delta s_1}{\mu_i} = T_Q(\Delta s_1),\ \forall i \in \mathcal{D}_1 \subset \bar{\mathcal{B}}_0, \tag{83}$$

i.e., some non-bottleneck clouds become bottleneck when S increases to $S_1^{(0)}$. Since Cloud Q must satisfy (83) when $\Delta s = \Delta s_1$, considering (83) with $i = Q$ in particular, we have

$$S_1^{(0)} = S_0 + \Delta s_1 = S_0 + \frac{\mu_Q l_0(K-Q) - l_0\sum_{i\in\mathcal{B}_0}\mu_i}{\lambda\sum_{i\in\mathcal{B}_0}\mu_i + \lambda\mu_Q(Q-J)}.$$

It can be shown that $S_1^{(0)}$ satisfies (54). As long as $S < S_1^{(0)}$, we have $\mathcal{B}_{\Delta s} = \mathcal{B}_0$ and $x = Q$, and $T^* = T_Q(\Delta s)$, which can be achieved by a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ if and only if it satisfies (82), or equivalently, (18).

Now it remains to show that $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ satisfying (18) and (57), with $x = Q$, is in the feasible region. It is straightforward to check that given such $(\boldsymbol{l}^*, \boldsymbol{r}^*)$, the constraints in the minimization problem (32) can be satisfied, *provided* that there exists non-negative δ_i's satisfying (58) and (59) for all $i \in \mathcal{B}_0$. Note that they must exist due to the following two reasons. First, we have

$$r_i^* = \mu_i T^* \leq \mu_{Q+1} T_0 \leq l_0,\ \ \forall i \in \mathcal{B}_0,$$

where the first inequality follows from the fact that $\mu_i \leq \mu_{Q+1}$ for all $i \in \mathcal{B}_0$ and $T^* \leq T_0$, and the second inequality follows from (40). It implies that there must exist non-negative δ_i such that $l_i^* = r_i^* + \delta_i \leq l_0 + \lambda\Delta s$ for all $i \in \mathcal{B}_0$. Second, when (58) is fulfilled, we can show by (55) and (18) that

$$\sum_{i\in\mathcal{B}_0} \delta_i \leq \sum_{i\in\mathcal{B}_0} (l_0 + \lambda\Delta s - r_i^*).$$

Combining the above two facts, there exists non-negative δ_i's that satisfy (58) and (59). This proves the existence of a feasible pair satisfying (18) and (57) for $S < S_1^{(0)}$.

Next, we consider the setting when $S = S_1^{(0)}$, at which Cloud i ($\forall i \in \mathcal{D}_1$) becomes a bottleneck cloud. This implies $x = Q - q_1$ and $\mathcal{B}_{\Delta s_1} = \mathcal{B}_0 \cup \mathcal{D}_1 = \{x+1, \ldots, V\}$. On the other hand, it is easy to see that the optimal computing time is the same as the case where $\Delta s = \Delta s_1 - \epsilon$, where $\epsilon \to 0^+$. When $\Delta s > \Delta s_1$, $\frac{r_i^*}{\mu_i} > T_Q(\Delta s)$ for all $i \in \mathcal{D}_1$, so $T_Q(\Delta s)$ is not achievable and we need to find another lower bound.

Consider the setting with $S_1^{(0)} \leq S < S_2^{(0)}$. We have $\mathcal{B}_{\Delta s} = \mathcal{B}_{\Delta s_1}$ and $x = Q - q_1 > J$, where $S_2^{(0)}$ is obtained by (54) like $S_1^{(0)}$. Similar to (47), we have

$$T \geq \frac{\sum\limits_{i\in\mathcal{B}_{\Delta s_1}} r_i}{\sum\limits_{i\in\mathcal{B}_{\Delta s_1}} \mu_i} \geq \frac{(K-x)l_0 - (x-J)\lambda\Delta s}{\sum_{i=x+1}^{V}\mu_i} \triangleq T_x(\Delta s).$$

The equality holds iff $r_i^* = \mu_i T_x(\Delta s)$ $\forall i \in \mathcal{B}_{\Delta s_1}$ and $r_i^* = l_i^* = l_0 + \lambda\Delta s$ $\forall i \in \bar{\mathcal{B}}_{\Delta s_1}$. For all $i \in \mathcal{B}_{\Delta s_1}$, let $l_i^* = r_i^* + \delta_i$, where $\delta_i > 0$ can be any number satisfying (58) and $l_0 + \lambda\Delta s \geq l_{x+1}^* \geq l_{x+2}^* \geq \cdots \geq l_V^*$. Then such l_i^* and r_i^* achieve $T^* = T_x(\Delta s)$ while satisfying all the constraints in problem (32). Note that T^* is continuous at $S_1^{(0)}$.

The same argument can be applied to cases where $S_i^{(0)} \leq S < S_{i+1}^{(0)}$, for $i = 1, 2, \ldots, \Pi - 1$. When $S \geq S_\Pi^{(0)}$, $x = J$. Same as in the proof of Proposition 16, T^* cannot be further reduced when S increases, which completes the proof for this case.

Case 2(a): $V - K > Q > J$.

When $x = Q$, Lemma 17 shows that $T_Q(\Delta s)$ is achieved by a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ if and only if it satisfies

$$l_i^* = \begin{cases} l_1^*, & i \in \{1,2,\ldots,x\} \\ \mu_i T^*, & i \in \mathcal{I} \triangleq \cup_{i\in\mathcal{N}}\mathcal{D}_i \\ l_V^*, & i \in \{x+1, x+2, \ldots, V\} \setminus \mathcal{I} \end{cases}, \tag{84}$$

$$r_i^* = \begin{cases} l_1^* & i \in \{1,2,\ldots,x\} \\ \mu_i T^* & i \in \{x+1, x+2, \ldots, V\} \end{cases}, \tag{85}$$

where $\mathcal{N} = \varnothing$. Now we will show that with $x = Q$, a pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ satisfying (84), (85) and

$$\begin{cases} \sum_{i=1}^{V} l_i^* = S \\ \sum_{i=V-K+1}^{V} l_i^* - \sum_{i=1}^{J} l_i^* = 1 \\ \sum_{i=V-K+1}^{V} l_i^* = \sum_{i=1}^{V} r_i^* \end{cases}, \tag{86}$$

is in the feasible region. By solving the linear equations, we obtain $l_1^* = l_0 + \lambda\Delta s$, $l_V^* = l_0 + \frac{J}{K}\lambda\Delta s$ and $T^* = T_Q(\Delta s)$. It is straightforward to check that such $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ is in the feasible region when $x = Q$. Similar as in Case 1, when S increases to $S_1^{(0)}$, some non-bottleneck clouds become bottleneck, i.e., x drops to $Q - q_1$ and $r_i^* = l_1^* = \mu_i T^*$, $\forall i \in \mathcal{D}_1$. Combining this condition with (84), (85) and (86), the value of $S_1^{(0)}$ can be determined. When $S_0 \leq S < S_1^{(0)}$ the optimal computing time $T^* = T_Q(\Delta s)$ and $x = Q$. When $S > S_1^{(0)}$, $x = Q - q_1$ and $T_Q(\Delta s)$ is not achievable because $l_i^* > l_V^*$ for $i \in \mathcal{D}_1 \subset \{x+1, x+2, \ldots, V\}$, so we need to find another lower bound.

Consider the setting with $S > S_1^{(0)}$ and $x = Q - q_1$. Since $\mu_i T \geq r_i$ for all $i \in \mathcal{V}$, we have

$$\begin{aligned} T &\overset{(a)}{\geq} \frac{\sum_{i\in\mathcal{B}_0} r_i}{\sum_{i\in\mathcal{B}_0} \mu_i} = \frac{\sum_{i\in\mathcal{V}} r_i - \sum_{i\in\bar{\mathcal{B}}_0} r_i}{\sum_{i\in\mathcal{B}_0} \mu_i} \\ &\overset{(b)}{\geq} \frac{\sum_{i=V-K+1}^{V} l_i - \sum_{i=1}^{x} l_i - \sum_{i\in\mathcal{D}_1} r_i}{\sum_{i\in\mathcal{B}_0} \mu_i} \\ &= \frac{(K-x)l_0 - \sum_{i=J+1}^{x} \Delta l_i - \sum_{i\in\mathcal{D}_1} r_i}{\sum_{i\in\mathcal{B}_0} \mu_i}, \end{aligned} \tag{87}$$

where (a) is tight if and only if $r_i^* = \mu_i T,\ i \in \mathcal{B}_0$, and (b) is tight if and only if $l_i^* = r_i^*,\ i \in \{1, 2, \ldots, x\}$. Define

$$\delta' \triangleq \sum_{i=1}^{x} \Delta l_i = \sum_{i=1}^{J} \Delta l_i + \sum_{i=J+1}^{x} \Delta l_i$$

like (77). Similar as the proof in Lemma 17, we can derive

$$\begin{aligned} \Delta s &\overset{(c)}{\geq} \delta' + \sum_{i\in\mathcal{D}_1} \Delta l_i + \sum_{i=Q+1}^{V-K} \Delta l_i + \sum_{i=V-K+1}^{V} \Delta l_i \\ &\overset{(d)}{\geq} \delta' + \sum_{i\in\mathcal{D}_1} \Delta l_i + \frac{J(V-Q)}{Kx}\delta' \\ &\overset{(e)}{\geq} (1 + \frac{J(V-Q)}{Kx})\delta' + \sum_{i\in\mathcal{D}_1} (r_i - l_0), \end{aligned} \tag{88}$$

where (c) is tight if and only if $\sum_{i\in\mathcal{V}} l_i^* = S$, (d) is tight if and only if $l_i^* = l_V^*\ \forall i \in \mathcal{B}_0$ and $l_i^* = l_1^*\ \forall i \in \{1, 2, \ldots, x\}$, and (e) is tight if and only if $r_i^* = l_i^*\ \forall i \in \mathcal{D}_1$. Furthermore,

$$\sum_{i\in\mathcal{D}_1} r_i \leq \sum_{i\in\mathcal{D}_1} \mu_i T \tag{89}$$

is tight if and only if $r_i^* = \mu_i T,\ \forall i \in \mathcal{D}_1$. Combining the inequalities (87), (88), (89) and constraints in (35), (38), we can derive a new lower bound for T, which is achieved by a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ if and only if it satisfies (84) and (85) hold with $\mathcal{N} = \{1\}$.

Now we will show that with $x = Q - q_1$, a pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ satisfying (84), (85) and (86) may not be in the feasible region when storage budget S becomes larger. As S increases, $\forall i \in \mathcal{D}_1$, $l_i^* = r_i^* = \mu_i T^*$ will decrease due to the dropping T^* and l_V^* will increase due to the growing S, so l_i^* has a chance to be smaller than l_V^*. If we combine (84), (85), (86) and

$$l_i^* = r_i^* = \mu_i T^* = l_V^*,\ \forall i \in \mathcal{D}_1, \tag{90}$$

$S_2^{(1)}$ can be determined. Once $S > S_2^{(1)}$, $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ is no longer in the feasible region. Meanwhile, when S increases to $S_2^{(0)}$, which is determined similar as $S_1^{(0)}$, Cloud i ($i \in \mathcal{D}_2$) will become bottleneck, i.e., x drops to $Q - q_2$ and $r_i^* = l_i^* = \mu_i T^*$, $\forall i \in \mathcal{D}_2$. If $S_2^{(1)} < S_1^{(0)}$ or $S_2^{(1)} > S_2^{(0)}$, then for $S_1^{(0)} < S < S_2^{(0)}$ a pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ satisfying (84), (85) and (86) is in the feasible region, as no constraints are violated. If $S_1^{(0)} \leq S_2^{(1)} \leq S_2^{(0)}$, the pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ is in the feasible region when $S_1^{(0)} < S \leq S_2^{(1)}$. For $S_2^{(1)} < S \leq S_2^{(0)}$, the lower bound for T becomes $T_Q(\Delta s)$, which can be achieved by a feasible pair $(\boldsymbol{l}^*, \boldsymbol{r}^*)$ satisfying (84) and (85) hold with $\mathcal{N} = \varnothing$. It should be noted that there is an additional turning point at $S = S_2^{(1)}$, since the optimal condition changes.

The same argument can be applied to cases where $S_i^{(0)} \leq S < S_{i+1}^{(0)}$, for $i = 2, \ldots, \Pi - 1$, which is concluded by Algorithm 2. Note that there can be at most one additional turning point within the interval $[S_1^{(0)}, S_2^{(0)})$, but there can be multiple additional turning points in later intervals. When $S_0 \leq S < S_\Pi^{(0)}$, (60) can be determined by solving (84), (85) and (86). When $S \geq S_\Pi^{(0)}$, $x = J$. Same as in the proof of Proposition 16, T^* cannot be further reduced when S increases, which completes the proof for this case.

Case 2(b): $Q \geq V - K > J$.

This case combines Case 1 and Case 2(a). If $x \geq V - K$, the optimal computing time and allocation scheme are just the same as Case 1, which can be proved in a similar manner. If $x < V - K$, the optimal computing time and allocation scheme can be obtained by Algorithm 2, whose proof is similar as Case 2(a). □

REFERENCES

[1] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, Feb. 2013.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[3] A. Severinson, A. G. I. Amat, and E. Rosnes, "Block-diagonal and LT codes for distributed computing with straggling servers," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 1739–1753, Mar. 2019.

[4] H. Park, K. Lee, J.-Y. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1630–1634.

[5] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2418–2422.

[6] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4403–4413.

[7] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1920–1933, Mar. 2020.

[8] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1988–1992.

[9] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed–Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2027–2031.

[10] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2018, pp. 857–866.

[11] B. Hasircioglu, J. Gómez-Vilardebó, and D. Gunduz, "Bivariate polynomial coding for straggler exploitation with heterogeneous workers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 251–256.

[12] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.

[13] M. Kiamari, C. Wang, and A. S. Avestimehr, "On heterogeneous coded distributed computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–7.
[14] D. Kim, H. Park, and J. K. Choi, "Optimal load allocation for coded distributed computation in heterogeneous clusters," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 44–58, Jan. 2021.
[15] M. Kim, J. Sohn, and J. Moon, "Coded matrix multiplication on a group-based model," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 722–726.
[16] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 34–40, Apr. 2017.
[17] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.
[18] Y. H. Ezzeldin, M. Karmoose, and C. Fragouli, "Communication vs distributed computation: An alternative trade-off curve," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2017, pp. 279–283.
[19] L. Song, C. Fragouli, and T. Zhao, "A pliable index coding approach to data shuffling," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1333–1353, Mar. 2020.
[20] M. A. Attia and R. Tandon, "Information theoretic limits of data shuffling for distributed learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[21] R. Bitar, P. Parag, and S. E. Rouayheb, "Minimizing latency for secure distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2900–2904.
[22] R. Bitar, P. Parag, and S. E. Rouayheb, "Minimizing latency for secure coded computing using secret sharing via staircase codes," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4609–4619, Aug. 2020.
[23] H. Yang and J. Lee, "Secure distributed computing with straggling servers using polynomial codes," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 1, pp. 141–150, Jan. 2019.
[24] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. NIPS Syst. ML Workshop*, 2018, pp. 1215–1225.
[25] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[26] J. Kakar, S. Ebadifar, and A. Sezgin, "On the capacity and straggler-robustness of distributed secure matrix multiplication," *IEEE Access*, vol. 7, pp. 45783–45799, 2019.
[27] Z. Jia and S. A. Jafar, "On the capacity of secure distributed batch matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 67, no. 11, pp. 7420–7437, Nov. 2021.
[28] R. G. L. D'Oliveira, S. E. Rouayheb, and D. Karpuk, "GASP codes for secure distributed matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 66, no. 7, pp. 4038–4050, Jul. 2020.
[29] G. L. R. D'Oliveira, S. E. Rouayheb, D. Heinlein, and D. Karpuk, "Degree tables for secure distributed matrix multiplication," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Aug. 2019, pp. 1–5.
[30] J. Kakar, A. Khristoforov, S. Ebadifar, and A. Sezgin, "Uplink cost adjustable schemes in secure distributed matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 1124–1129.
[31] Z. Chen, Z. Jia, Z. Wang, and S. A. Jafar, "GCSA codes with noise alignment for secure coded multi-party batch matrix multiplication," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 306–316, Mar. 2021.
[32] H. Akbari-Nodehi and M. A. Maddah-Ali, "Secure coded multi-party computation for massive matrix operations," *IEEE Trans. Inf. Theory*, vol. 67, no. 4, pp. 2379–2398, Apr. 2021.
[33] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci.*, vol. 305, pp. 357–383, Jun. 2015.
[34] Z. Li, M. Liang, L. O'Brien, and H. Zhang, "The cloud's cloudy moment: A systematic survey of public cloud service outage," *Int. J. Cloud Comput. Services Sci.*, vol. 2, no. 5, pp. 321–331, Dec. 2013.
[35] P. Hu, C. W. Sung, S.-W. Ho, and T. H. Chan, "Optimal coding and allocation for perfect secrecy in multiple clouds," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 388–399, Feb. 2016.
[36] J. Chen, C. W. Sung, and T. H. Chan, "Storage and computation: A tradeoff in secure distributed computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.
[37] A. Subramanian and S. W. McLaughlin, "MDS codes on the erasure-erasure wiretap channel," 2009, *arXiv:0902.3286*.
[38] S. Pawar, S. E. Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6734–6753, Oct. 2011.
[39] M. Dai, Z. Zheng, S. Zhang, H. Wang, and X. Lin, "SAZD: A low computational load coded distributed computing framework for IoT systems," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3640–3649, Apr. 2020.
[40] C. W. Sung, K. W. Shum, Q. Yu, and G. Xu, "Maximally recoverable codes: Connections to generic network coding and maximal matching," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2017, pp. 36–40.
[41] R. W. Yeung, *Information Theory and Network Coding*. New York, NY, USA: Springer-Verlag, 2008.
[42] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North Holland, 1977.
[43] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA, USA: SIAM, 2000.
[44] R. K. Sundaram, *A First Course in Optimization Theory*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[45] A. W. Marshall, I. Olkin, and B. C. Arnold, *Inequalities: Theory of Majorization and Its Applications*. Academic, 1979.
[46] *MPICH: High-Performance Portable MPI*. Accessed: Oct. 6, 2020. [Online]. Available: https://www.mpich.org

Jiajun Chen received the B.S. degree from Sichuan University, Chengdu, China, in 2018. She is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong. Her research interests include coded distributed computing, information theory, and resource allocation.

Chi Wan Sung received the B.Eng., M.Phil., and Ph.D. degrees in information engineering from The Chinese University of Hong Kong in 1993, 1995, and 1998, respectively. He worked as an Assistant Professor at The Chinese University of Hong Kong in 1999, and then joined the Faculty at the City University of Hong Kong in 2000. He is currently an Associate Professor and the Associate Head (Undergraduate Programs) of the Department of Electrical Engineering. His current research interests include interplay between coding, communications, and computing, with emphasis on algorithm design and complexity analysis. He was an Associate Editor of the *Transactions on Emerging Telecommunications Technologies* (ETT) from 2013 to 2016. He is on the Editorial Boards of *ETRI Journal* and *Electronics Letters*.

Terence H. Chan received the Ph.D. degree in February 2001. He was an Assistant Professor with The Chinese University of Hong Kong in 2001. From February 2002 to June 2004, he was a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto. In 2004, he became an Assistant Professor with the Department of Computer Science, University of Regina, Canada. He joined the Institute for Telecommunications Research (ITR), University of South Australia, as a Senior Research Fellow, in 2006. He is currently an Associate Professor with the ITR.