Decode-and-Compare: An Efficient Verification Scheme for Coded Edge Computing

Mingjia Fu[1], Jin Wang[1⋆], Jianping Wang[2], Kejie Lu[3], Admela Jukan[4] and Fei Gu[1]
[1] School of Computer Science and Technology, Soochow University, P.R.China
[1] Collaborative Innovation Center of Novel Software Technology and Industrialization
[2] Department of Computer Science, City University of Hong Kong
[3] Department of Computer Science and Engineering, University of Puerto Rico at Mayagüez
[4] Technische Universität Carolo-Wilhelmina zu Braunschweig, Germany

***Abstract*—Edge computing is a promising technology that can fulfill the requirements of latency-critical and computation-intensive applications. To further enhance the performance, coded edge computing has emerged because it can optimally utilize edge devices to speed up the computation. In this paper, we tackle a major security issue in coded edge computing: how to verify the correctness of results and identify attackers. Specifically, we propose an efficient verification scheme, namely *Decode-and-Compare* (DC), by leveraging both the coding redundancy of edge devices and the properties of linear coding itself. To design the DC scheme, we conduct a solid theoretical analysis to show the required coding redundancy, the expected number of decoding operations, and the tradeoff between them. To evaluate the performance of DC, we conduct extensive simulation experiments and the results confirm that the DC scheme can outperform existing solutions, such as homomorphic encryption and computing locally at the user device.**

I. Introduction

Edge computing can utilize computing facilities at the edge of networks to support latency-critical and computation-intensive applications, such as big data analytics, machine learning and virtual/augmented/mixed reality (VR/AR/MR) [1]. However, in edge computing, since edge devices are less reliable than typical computing devices in a cloud data center, they may not return the desired results on time. To address this issue, *coded edge computing* has been proposed to solve the computing stragglers by leveraging coding redundancy, in which the edge devices operate directly on coded data and send the intermediate results to the user [2]–[4].

In addition to the computing delay issue, another main concern is security. Since edge devices may not be trustworthy, coded computing schemes have been designed to provide confidentiality of the computing data such that edge devices cannot obtain the information from the coded data [5]–[8].

⋆Corresponding author: Jin Wang, wjin1985@suda.edu.cn.

This work was supported in part by the National Natural Science Foundation of China (No.61672370), National Science Foundation (NSF) under grant CNS-1730325, "Six Talent Peak" Project of Jiangsu Province (No. XYDXX-084), Hong Kong Research Grant Council (No. GRF 11211519), Science Technology and Innovation Committee of Shenzhen Municipality (No. JCYJ20170818095109386), Tang Scholar of Soochow University, the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) and Postgraduate Research & Practice Innovation Program of Jiangsu Province (No. SJCX18_0845, SJCX19_0800).

Besides confidentiality, verifiable computing has always been an important topic in outsourced computing or distributed computing area. In coded computing, a single modified intermediate result can screw up the final result. However, the verifiable computation problem of coded computing has not been fully investigated. Specifically, most existing verification schemes for outsourced computation are based on interactive proof, computation replication or homomorphic encryption [9]–[15]. Firstly, interactive proof schemes are based on a garbled circuit [9]. The pre-processing process of the circuit design is directly proportional to the function complexity and scale [10]. Secondly, most computation replication schemes are designed to detect Byzantine faults via certain consensus schemes, which lead to *high computational overhead*. For instance, to tolerate t faults for one computation task, a typical scheme is $3t$ replication* [11]–[13]. Thirdly, to verify computing results, homomorphic encryption schemes can be applied, which not only need a *key distribution infrastructure* to share the keys [10], [14], but also have *very high computation complexity* [10], [14], [15].

To enhance the security and reliability of coded edge computing, in this paper, we will focus on a verification problem, *i.e.*, how to verify the correctness of computing results and then further identify the set of faulty edge devices (attackers) that return incorrect results. In our study, we will take matrix multiplication, which is a fundamental building block of many distributed machine learning algorithms, as the representative computation task [2]–[7].

For the verification problem, we propose an efficient verification scheme, namely, *Decode-and-Compare* (DC), by leveraging the coding redundancy from edge devices and the properties of linear coding itself without sharing any encryption key. Certainly, if no edge devices return incorrect intermediate results, then the same final result can be obtained by different sets of k intermediate results. On the other hand, if one edge device returns an incorrect intermediate result, with high probability, a certain subset of final results will be incorrect and appear differently. Based on the main idea, we will design the DC verification scheme in this paper. Firstly, compared with the traditional replication verification schemes,

*This case is shown in Remark 1 of Theorem 1 when $k = 1$ and $\beta = \frac{1}{3}$.

to ensure the correctness of the computation result, the lower bound of the DC scheme redundancy rate can be reduced. Secondly, including the integrity, the confidentiality of the data can also be provided, because the proposed DC scheme can well compatible with the existing secure coded distributed computing scheme. Thirdly, with sufficient coding redundancy, it is possible to not only distinguish the correct final result from incorrect ones, but also identify the attackers or nodes with abnormal behavior at the cost of higher verification complexity. In this paper, we will systematically investigate these issues in the design of the DC verification scheme. Specifically, we will analyze the relationship between the coding redundancy and the verification complexity.

The rest of the paper is organized as follows. First, we give the system model and the attack model in Sec. II. Next, we present the design of the DC verification scheme in Sec. III-A, and then conduct solid theoretical analysis in Sec. III-B. Finally, in Sec. IV, we conduct extensive simulation experiments, before concluding the paper in Sec. V.

II. Problem Modeling

In this section, we will introduce the coded edge computing model and the attack model considered in this paper.

A. System Model

In this paper, we focus on the coded edge computing and we take matrix multiplication as an example [2]–[7]. Specifically, we consider the same computation model as studied in [4], [6], [17]–[21], in which a user device wants to obtain the result $\mathbf{Y}$ by multiplying a data matrix $\mathbf{A} \in \mathbb{F}_q^{m\times n}$, *e.g.*, a trained model stored in the cloud, with an input matrix $\mathbf{X} \in \mathbb{F}_q^{n\times l}$ for a sufficiently large finite field $\mathbb{F}_q$.

To utilize the distributed computation power of edge devices, the cloud encodes data matrix $\mathbf{A}$ and distributes the coded data blocks to edge devices. Specifically, it first equally divides $\mathbf{A}$ by k row partitions, *i.e.*, $\mathbf{A}$ can be expressed as block matrix $\mathbf{A} = [\mathbf{A}_1^\top, \cdots, \mathbf{A}_k^\top]^\top$, $1 \le k \le m$, in which each row partition $\mathbf{A}_i$ has dimension $\frac{m}{k} \times n$. When $k = 1$, the scheme becomes computation replication. Next, the cloud can determine an encoding matrix [2]–[7], [17]–[20], *i.e.*, the $k' \times k$ dimensional matrix $\mathbf{B} = [\mathbf{B}_1^\top, \cdots, \mathbf{B}_{k'}^\top]^\top$, where $k' \ge k$. We note that the proposed DC verification scheme is compatible with most of existing coded distributed computing system [2]–[7], [17]–[20], because it does not need to modify the existing coded computing schemes including the coding matrix $\mathbf{B}$.

Definition 1. ***(Block matrix multiplication $\circledast$)*** *Assume that $b_{i,j}$ is the element located in the i-th row j-th column of matrix $\mathbf{B}$, the operator $\circledast$ is defined as follows:*

$$\mathbf{P}_i = \mathbf{B}_i \circledast \mathbf{A} = \sum_{j=1}^{k} \mathrm{b}_{i,j}\mathbf{A}_j, \forall i \in \{1, \cdots, k'\},$$
$$\mathbf{P} = \mathbf{B} \circledast \mathbf{A} = [\mathbf{P}_1^\top, \cdots, \mathbf{P}_{k'}^\top]^\top.$$

The cloud then encodes the data matrix $\mathbf{A}$ into k' coded blocks: $\mathbf{P} = [{\mathbf{P}_1}^\top, \cdots, {\mathbf{P}_{k'}}^\top]^\top = \mathbf{B} \circledast \mathbf{A}$, in which each $\mathbf{P}_i = \mathbf{B}_i \circledast \mathbf{A}$ is allocated to an edge device, $\forall i \in \{1, \cdots, k'\}$, and $\mathbf{B}_i$ is called as the *encoding vector* of $\mathbf{P}_i$ [2]–[7], [17]–[20]. Let $\alpha = \frac{k'}{k}$ be the *coding redundancy ratio*. We note that such data encoding and distribution process is only needed to be done once for the computation tasks of multiplying the same data matrix $\mathbf{A}$ with different input matrices. Next, we explain the computing process. Without loss of generality, we assume that $\mathbf{P}_i$ is allocated to edge device w_i, $\forall i \in \{1, \cdots, k'\}$. After the user device uploads the input matrix $\mathbf{X}$ to all the involved edge devices, each edge device w_i computes the *intermediate results* $\mathbf{I}_i = \mathbf{P}_i\mathbf{X}$, $\forall i \in \{1, \cdots, k'\}$ and sends it to the user.

Finally, we explain the decoding process. Here we first note that, to exploit the diversity of the coded data blocks and intermediate results in edge devices, the cloud can determine $\mathbf{B}$ such that every k rows of $\mathbf{B}$ can form a full rank $k \times k$ matrix [2]–[7], [17]. In this manner, as soon as the user device receives k intermediate results, it can decode and recover the *final result* $\mathbf{Y} = \mathbf{AX}$. Specifically, suppose that $\mathbf{I}'$ is the matrix composed by k intermediate results as its row partitions and $\mathbf{I}' = (\mathbf{B}' \circledast \mathbf{A})\mathbf{X} = \mathbf{B}' \circledast \mathbf{Y}$, in which matrix $\mathbf{B}'$ is composed by the k encoding vectors of the k coded blocks for computing the k intermediate results. Since every k encoding vectors in $\mathbf{B}$ can form a full rank matrix, in the case that $\mathbf{B}$ is already known, once receiving k intermediate results, the user can decode and obtain the final result $\mathbf{Y} = {\mathbf{B}'}^{-1} \circledast \mathbf{I}'$.

B. Attack Model

In this paper, we focus on the *pollution attacks* in which each intermediate result $\mathbf{I}_i$ may be modified by a malicious edge device or an incorrect intermediate result is given by a faulty edge device. Specifically, in this paper, we study the case that these edge devices cannot collude with each other.

Let β be the attacker ratio, $\beta > 0$, *i.e.*, there are $\beta k'$ edge devices being attackers and thus $\beta k'$ intermediate results are incorrect. In the non-collusion case, we consider the attack model that each intermediate result may be independently and randomly modified by each edge device to any matrix with the same dimension.

III. An Efficient Verification Scheme: Decode-and-Compare (DC)

In this section, we firstly give the design of the DC verification scheme. We then conduct a solid theoretical analysis to show the required coding redundancy and the number of decoding rounds to obtain the correct final result.

A. The Main Idea

In this subsection, we describe the main ideas of the DC verification scheme.

Let $\mathbf{I}$ be the set of intermediate results $\{\mathbf{I}_1, \cdots, \mathbf{I}_{k'}\}$ received by the user device. Let $\mathbf{I}^\vee$ be the set of correct intermediate results and $\mathbf{I}^\times$ be the set of incorrect intermediate results. We have $\mathbf{I} = \mathbf{I}^\vee \cup \mathbf{I}^\times$ and $\mathbf{I}^\vee \cap \mathbf{I}^\times = \emptyset$. To verify the final result, the main idea of our DC verification scheme is shown as follows: When $k' > k$, the user can decode and obtain multiple final results according to different groups of k intermediate results, called *verification groups*. If all the

intermediate results in two verification groups have not been maliciously modified, then the two final results obtained based on them must be the same. Otherwise, since each intermediate result may be independently and randomly modified by each edge device, the obtained final results are different with *very high probability*†. Therefore, we repeatedly select different verification groups from $\mathbf{I}$ to **decode** the final results and then **compare** them to determine the correct final result. Once we obtain two identical final results, we can determine that this final result is correct. Since "decode" and "compare" are the main procedures in our scheme, we name it as **decode-and-compare** verification scheme. The details of the algorithm to obtain the correct final result are shown in Algorithm 1.

The DC verification scheme may not successfully verify the correct final result in the following two cases: (1) the number of correct intermediate results $|\mathbf{I}^{\surd}| \leq k$, *i.e.* $|\mathbf{I}^{\times}| \geq k' - k$, because at least two groups of k correct intermediate results are needed to decode and compare, or (2) two incorrect final results are the same. For the first case, since $|\mathbf{I}^{\surd}| = (1-\beta)k' = (1-\beta)\alpha k$, when given β and k', we can reduce the number of blocks divided from the data matrix $\mathbf{A}$, *i.e.*, k, to make sure $|\mathbf{I}^{\surd}| > k$. When given β and k, we can increase the coding redundancy ratio, *i.e.*, α, to make sure $|\mathbf{I}^{\surd}| > k$. For the second case, the probability will approach to 0, when the size of matrices, *e.g.*, $\mathbf{A}$ and $\mathbf{X}$, and the size of the finite field are sufficiently large. We note that the proposed DC verification scheme does not need to modify the existing coded computing schemes [2]–[7], [17]–[20] except the verification on the user. After the user obtains the correct final result, with sufficient coding redundancy, it is possible to distinguish the incorrect intermediate results and further identify the attackers or nodes with abnormal behavior at the cost of higher verification complexity. Due to the space limit, we will discuss this issue in our future work.

To make sure the obtained result is correct, the user can locally computes $\mathbf{AX}$. In this case, the computational complexity is $O(mnl)$. In the proposed DC verification scheme, to verify the correct final result, the computational complexity on the user device is $O(N_\eta(k^3+kml))$, in which N_η is the decoding rounds and $O(k^3+kml)$ is the computational complexity of a single round.

We note that the decoding complexity to verify the correct final result is irrelevant to the column size of $\mathbf{A}$, *i.e.*, n. The number of blocks divided from data matrix $\mathbf{A}$, *i.e.*, k, is less than the number of involved edge nodes, *i.e.*, k', which is at most in tens in practice. Clearly, compared with m and n, which may be tens of thousands in the data matrix [2], [22], [23], k is much less than n. Therefore, compared with centralized computation at the user, the proposed DC verification scheme can significant reduce computation time when m and n are very large. Moreover, N_η is highly related to the coding redundancy ratio α, the attacker ratio β, and the number of blocks k. We will show in Theorem 3 that N_η can be reduced by adjusting the parameters k and α even when the attacker ratio β is sufficiently large.

†The probability will approach to 1, when the size of matrices, *e.g.*, $\mathbf{A}$ and $\mathbf{X}$, and the size of the finite field are sufficiently large. In practice, the parameters m, l and q for machine learning based applications are usually very large. Due to the space limit, we will formally present the theoretical analysis of the probability in our future work.

Algorithm 1: DC Verification Scheme Part 1: Final Result Verification

Input: $\mathbf{I}, \mathbf{B}, k, k'$
Output: $\mathbf{Y}, \mathbf{I}^Y$
$\bar{\mathbf{I}} = \emptyset, \overline{\mathbf{Y}} = \emptyset, i = 1$;
while $i \leq \binom{k'}{k}$ **do**
  Select a new verification group $\bar{\mathbf{I}}_i = \{\mathbf{I}_{c_1}, \cdots, \mathbf{I}_{c_k}\} \notin \bar{\mathbf{I}}$, in which $\mathbf{I}_{c_h} \in \mathbf{I}, \forall h \in \{1, \cdots, k\}$;
  $\bar{\mathbf{I}} = \bar{\mathbf{I}} \cup \{\bar{\mathbf{I}}_i\}$;
  $\mathbf{B}'_i = [\mathbf{B}_{c_1}^\top, \cdots, \mathbf{B}_{c_k}^\top]^\top$;
  $\mathbf{I}'_i = [\mathbf{I}_{c_1}^\top, \cdots, \mathbf{I}_{c_k}^\top]^\top$;
  $\mathbf{Y}_i = \mathbf{B}_i'^{-1} \circledast \mathbf{I}'_i$; //block matrix multiplication;
  if $\exists \boldsymbol{Y}_j \in \overline{\boldsymbol{Y}}, \boldsymbol{Y}_i = \boldsymbol{Y}_j$ **then**
    $\mathbf{Y} = \mathbf{Y}_i$;
    $\mathbf{I}^Y = \bar{\mathbf{I}}_i \cup \bar{\mathbf{I}}_j$;
    return $\mathbf{Y}$ and $\mathbf{I}^Y$;
  else
    $\overline{\mathbf{Y}} = \overline{\mathbf{Y}} \cup \{\mathbf{Y}_i\}$;
    $i = i + 1$;
return false;

B. Theoretical Analysis of the DC Verification Scheme

In this subsection, we show the theoretical analysis of the DC verification scheme. We firstly derive the sufficient and necessary condition that the DC verification scheme can distinguish and obtain the correct final result on the required coding redundancy ratio.

Theorem 1. *The DC verification scheme can distinguish and obtain the correct final result, iff coding redundancy ratio $\alpha \geq \frac{k+1}{(1-\beta)k}$, in which β is the attacker ratio.*

Proof. If the user can distinguish the correct final result, there exist at least two groups of k correct intermediate results. Therefore, there are at least $k+1$ correct intermediate results, *i.e.*, $(1-\beta)k' \geq k+1$. Since $k' = \alpha k$, we have $(1-\beta)\alpha k \geq k+1$, *i.e.*, $\alpha \geq \frac{k+1}{(1-\beta)k}$.

If the coding redundancy ratio $\alpha = \frac{k'}{k} \geq \frac{k+1}{(1-\beta)k}$, we have the number of correct intermediate results $(1-\beta)k' = (1-\beta)\alpha k \geq k+1$. Therefore, there exist two groups of k correct intermediate results, based on which, the user can find that the two obtained final results are the same and figure out the correct final result. □

Remark 1. *From the above theorem, we know that (1) compared with computation replication verification scheme (the same as the DC verification when $k = 1$), the required redundancy ratio can be reduced from $\frac{2}{(1-\beta)}$ to $\frac{k+1}{(1-\beta)k}$; (2) when the attacker ratio β is large, high coding redundancy can be used to make sure the DC verification can find the correct final result; (3) when the data matrix is divided into more blocks, the required coding redundancy can be reduced.*

Next, the following two theorems show the lower bound and the upper bound of N_η, *i.e.*, the rounds of decoding to determine the correct results.

Theorem 2. $2 \leq N_\eta \leq \min\left(\binom{k'}{k} - \binom{v}{k} + 2, \binom{k'}{k}\right)$, *in which* $v = |\boldsymbol{I}^\vee| = (1-\beta)\alpha k$.

Proof. Firstly, to determine if the decoding result is correct, the user needs at least two decodings for comparison, *i.e.*, $N_\eta \geq 2$. Specifically, in the optimal case that the two verification groups in the first two decodings do not contain incorrect intermediate results, *i.e.*, $(\bar{\mathbf{I}}_1 \cup \bar{\mathbf{I}}_2) \subseteq \mathbf{I}^\vee$ and $N_\eta = 2$.

When $v \geq k+1$, *i.e.*, $\alpha \geq \frac{k+1}{(1-\beta)k}$, in the worst case, before the user finds two correct final results, it has decoded all the verification groups, each of which contains at least one incorrect intermediate result, *i.e.*, the rounds of decoding is $\binom{k'}{k} - \binom{v}{k}$. It also performs two additional rounds of decoding on two verification groups, each of which only contains correct intermediate results. Therefore, in the worst case, the number of decodings is $\binom{k'}{k} - \binom{v}{k} + 2$.

On the other hand, when $v \leq k$, there do not exist two verification groups, each of which only contains correct intermediate results, which is not sufficient to compared and distinguished the correct final result. In this case, the number of decoding rounds is $N_\eta = \binom{k'}{k}$. □

Remark 2. *From Theorem 2, when $v \leq k$, the DC verification scheme cannot determine the correct final result and the number of decoding rounds is $N_\eta = \binom{k'}{k}$. On the other hand, when $v \geq k+1$, the correct intermediate result can be verified, the number of decoding rounds $N_\eta \leq \binom{k'}{k} - \binom{v}{k} + 2$.*

Finally, we analyze the expected decoding rounds in the DC verification scheme when $v \geq k+1$, *i.e.*, $\alpha \geq \frac{k+1}{(1-\beta)k}$.

Theorem 3. *When $v \geq k+1$, the expected number of decoding rounds in the DC verification scheme is*

$$E(N_\eta) = \sum_{i=2}^{\binom{k'}{k}-\binom{v}{k}+2} \left(i(i-1) \frac{\binom{v}{k}}{\binom{k'}{k}} \prod_{j=0}^{i-3} \left(\frac{\binom{k'}{k} - \binom{v}{k} - j}{\binom{k'}{k} - j - 1} \right) \frac{\binom{v}{k} - 1}{\binom{k'}{k} - i + 1} \right).$$

Proof. If the DC verification scheme requires i rounds of decoding, the first $i-1$ rounds of decoding must obtain only one correct final result and the i-th round of decoding obtains another correct final result. As a correct result can be at any round of decoding. There are $i-1$ possible cases.

The probability that a correct final result is obtained in a particular number of decoding rounds is $\frac{\binom{v}{k}}{\binom{k'}{k}}$. The probability that the final results obtained from other $i-2$ rounds of decoding are incorrect final results is $\prod_{j=1}^{i-2} \left(\frac{\binom{k'}{k} - \binom{v}{k} - j + 1}{\binom{k'}{k} - j} \right)$ which equals to $\prod_{j=0}^{i-3} \left(\frac{\binom{k'}{k} - \binom{v}{k} - j}{\binom{k'}{k} - j - 1} \right)$. The probability that the last decoding obtains a correct result is $\frac{\binom{v}{k} - 1}{\binom{k'}{k} - i + 1}$. From Theorem 2, if $v \geq k+1$, $2 \leq i \leq \binom{k'}{k} - \binom{v}{k} + 2$. We have

$$E(N_\eta) = \sum_{i=2}^{\binom{k'}{k}-\binom{v}{k}+2} \left(i(i-1) \frac{\binom{v}{k}}{\binom{k'}{k}} \prod_{j=0}^{i-3} \left(\frac{\binom{k'}{k} - \binom{v}{k} - j}{\binom{k'}{k} - j - 1} \right) \frac{\binom{v}{k} - 1}{\binom{k'}{k} - i + 1} \right).$$

□

Remark 3. *Theorem 3 shows the expected number of decoding rounds in the DC verification scheme. We note that there exists a tradeoff between the coding redundancy ratio α and the expected number of decoding rounds N_η, i.e., a higher α, leading to the higher total computational overheads on edge devices, will reduce N_η, leading to the lower verification complexity at the user.*

IV. Simulations

In this section, we conduct two groups of experiments. To compare with the homomorphic encryption based verification schemes, we use Helib homomorphic encryption library developed by IBM [24]. We note that the proposed DC verification scheme does not need to modify the existing coded computing schemes [2]–[7], [17]–[20] except the verification on the user. Therefore, in this experiment, we only focus on the verification process of the user. Specifically, we perform matrix multiplication involved in the coded computing based on C++. The schemes are evaluated on a computer with Intel Core i5-6400 CPU 2.71GHz.

First, we compare the time consumptions of (1) the *homomorphic decryption* on the final result $\mathbf{Y}$, which is a key component of the homomorphic encryption/decryption based verification schemes, (2) the *centralized computation*, which means the user locally computes $\mathbf{AX}$ instead of outsourcing computation, and (3) the *decode-and-compare* verification scheme with different parameters related to matrix size. Then, we show the time consumptions of the *centralized computation* and the *decode-and-compare* verification scheme.

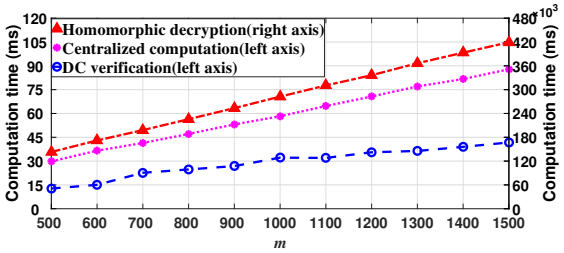
In this simulation, we consider six parameters as follows.

- m: the number of rows of matrices $\mathbf{A}$ and $\mathbf{Y}$.
- n: the number of rows of matrix $\mathbf{X}$.
- l: the number of columns of matrices $\mathbf{X}$ and $\mathbf{Y}$.
- α: coding redundancy ratio, $\alpha = \frac{k'}{k}$.
- β: attacker ratio, *i.e.*, there are $\beta k'$ attackers.
- k': the number of involved edge devices, *i.e.*, the total number of received intermediate results.

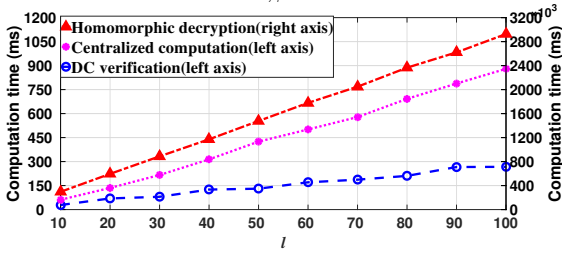
We note that in the DC verification scheme, the user needs to process N_η rounds of decoding on N_η verification groups to find the correct final result. It means that the time consumption of the DC verification scheme is N_η times that of coded edge computing without verification consideration. Therefore, in this experiment, instead of comparing with the coded edge computing without verification consideration, we show the rounds of decoding, *i.e.*, N_η, in Fig. 2.

A. Simulation results

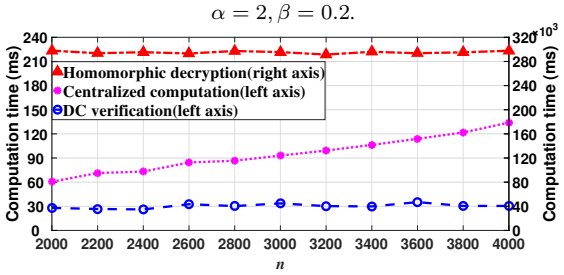
In the first group of the simulations, we compare the time consumptions of homomorphic decryption, centralized computation, and the DC verification when the computation scale parameters m, n, l continuously increase. As shown in Fig. 1, homomorphic decryption takes the longest time, which is 10^3 times higher than the other two schemes. Therefore, we use the axis on the right to express its time consumption.



a) Computation time (ms) when $n = 2000, l = 10, k' = 20, \alpha = 2, \beta = 0.2$.

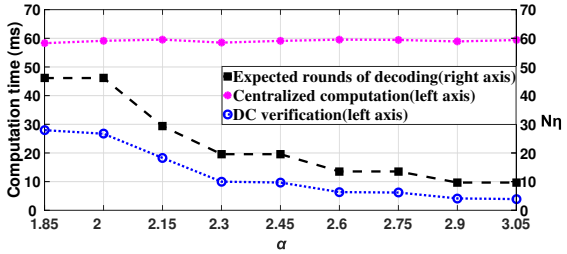


b) Computation time (ms) when $m = 1000, n = 2000, k' = 20, \alpha = 2, \beta = 0.2$.

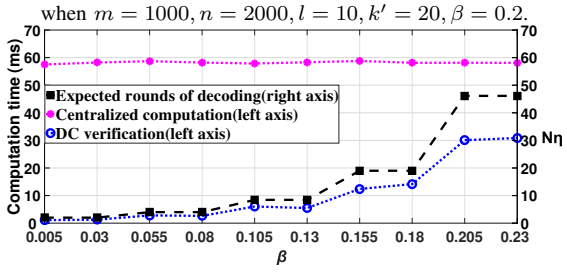


c) Computation time (ms) when $m = 1000, l = 10, k' = 20, \alpha = 2, \beta = 0.2$.

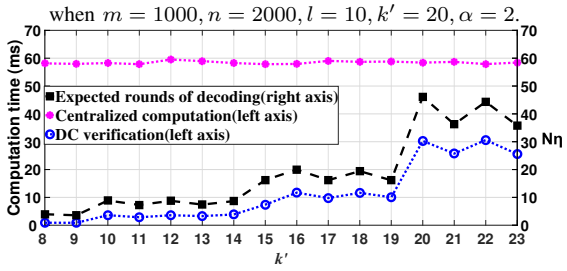
Fig. 1. The time consumptions of homomorphic decryption, centralized computation and DC verification schemes when changing the matrix size parameters m, l, n.



a) Computation time (ms) and expected rounds of decoding N_η when $m = 1000, n = 2000, l = 10, k' = 20, \beta = 0.2$.



b) Computation time (ms) and expected rounds of decoding N_η when $m = 1000, n = 2000, l = 10, k' = 20, \alpha = 2$.



c) Computation time (ms) and expected rounds of decoding N_η when $m = 1000, n = 2000, l = 10, \alpha = 2, \beta = 0.2$.

Fig. 2. The time consumptions of centralized computation and DC verification schemes, and expected rounds of decoding N_η when changing system parameters α, β and k'.

As shown in Fig. 1 a) and b), the time consumptions of the three schemes increase with the increases of m and l. According to the computation complexity analysis shown in Sec. III-A, the ratio between the computation complexities of DC verification scheme and the centralized computation is $N_\eta k/n$. According to Theorem 3, N_η is only related to k, k' and v. Since $k = k'/\alpha$ and $v = (1-\beta)k'$. Therefore, in Fig. 1 a) and b), the ratio between the time consumptions of them is almost the same when m and l increase because the parameter k', α and β are fixed. However, the gap between them increases with the increase of m and l because the gap between the computation complexities of them increases.

In Fig. 1 c), the time consumptions of homomorphic decryption and DC verification scheme almost do not change with the increase of n. The reason is that the computational complexity of homomorphic decryption is only related to the size of $\mathbf{Y}$, *i.e.*, $m \times l$. According to the computation complexity analysis shown in Sec. III-A, the computation complexity of DC verification scheme is also irrelevant to n while that of the centralized computation increases with the increase of n.

In the second group of the simulations, we compare the time consumptions of the centralized computation and the DC verification scheme when the system parameters α, β and k' increase. We use the right axis to represent the expected number of decoding rounds in the DC verification scheme.

In Fig. 2 a)-c), the time consumption of centralized computation is almost the same because the user locally computes $\mathbf{AX}$ in the centralized computation scheme, which is not affected by α, β and k'. On the other hand, in Fig. 2 a), the time consumption of the DC verification decreases with the increase of α. According to Theorem 3, the rounds of decoding, *i.e.*, N_η, decreases with the increase of α. Moreover, when k' and β are fixed and α increases, k decreases. Therefore, according to the verification complexity analysis shown in Sec. III-A, the computation time (complexity) of the DC verification decreases with the increase of α.

In Fig. 2 b), the time consumption of the DC verification increases with the increase of β because more attackers involved in the system leads to larger rounds of decoding, *i.e.*, N_η. In Fig. 2 c), the time consumption of the DC verification also increases with the increase of k'. The reason is that when α and β are fixed and k' increases, k increases. As opposed to the reason of Fig. 2 a), in this case, both N_η and the time consumption of the DC verification increases.

B. Simulation Conclusion

From above simulations, we have the following conclusions.

- The computation time of homomorphic decryption is about 10^3 times higher than the other schemes, which

shows that it is not suitable for large-scale computation tasks in edge computing for real-time applications.

- The proposed DC verification scheme can achieve a good performance when the number of involved edge devices $k' \leq 20$, coding redundancy ratio $\alpha \geq 2$, attacker ratio $\beta \leq 0.2$. As shown in Fig. 2 b), even when 23% edge devices are attackers, the computation time of the proposed DC verification scheme is more than 45% lower than that of centralized computation, *i.e.*, the user locally computes $\mathbf{AX}$. Moreover, according to Theorem 3, in practice, we can reduce N_η by adjusting the parameters k and α even when the attacker ratio β is sufficiently large.
- Although the user can locally computes $\mathbf{AX}$ to make sure the obtained result is correct, the computation complexity is directly proportional to the parameter n, *i.e.*, the number of columns of data matrix $\mathbf{A}$. However, n has no impact on the computation time (verification complexity) of the user when using the proposed DC verification scheme. Therefore, compared with the centralized computation, the proposed DC verification scheme can significantly reduce computation time when n is very large.
- In the proposed DC verification scheme, once the system can identify the attackers, it will not further allocate computation tasks to them, which will decrease the attacker ratio and further reduce the computation time (complexity) of DC verification scheme.

V. Conclusion

In this paper, we have studied the problem of verifiable coded edge computing, by leveraging coding redundancy of edge devices and the properties of linear coding itself, under the case that the intermediate results can be maliciously modified by edge devices. Specifically, we proposed an efficient DC verification scheme. We also conducted a solid theoretical analysis to show the rounds of decoding to obtain the correct final result. Finally, we conducted extensive simulation experiments to show the effectiveness of the proposed scheme.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, "Edge Computing: Vision and Challenges", *IEEE Internet of Things Journal (IOT)*, vol. 3, no. 5, pp. 637-646, 2016.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding Up Distributed Machine Learning Using Codes", *IEEE Transactions on Information Theory (TIT)*, vol. 64, no. 3, pp. 1514-1529, 2018.

[3] S. Li, M. A. Maddah-Ali, Q. Yu, A. S. Avestimehr, "A Fundamental Tradeoff between Computation and Communication in Distributed Computing", *IEEE Transactions on Information Theory (TIT)*, vol. 64, no. 1, pp. 109-128, 2018.

[4] Q. Yu, M. A. Maddah-Ali, A. S. Avestimehr, "Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding", in *Proc. of International Symposium on Information Theory (ISIT)*, pp. 2022-2026, 2018.

[5] R. Bitar, P. Parag, S. E. Rouayheb, "Minimizing Latency for Secure Distributed Computing", in *Proc. of International Symposium on Information Theory (ISIT)*, pp. 2900-2904, 2017.

[6] H. Yang, J. Lee, "Secure Distributed Computing with Straggling Servers Using Polynomial Codes", *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 14, no. 1, pp. 141-150, 2019.

[7] C. Cao, J. Wang, J. Wang, K. Lu, J. Zhou, A. Jukan and W. Zhao, "Optimal Task Allocation and Coding Design for Secure Coded Edge Computing", in *Proc. of the 39th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 1083-1093, 2019.

[8] R. Zhao, J. Wang, K. Lu, J. Wang, X. Wang, J. Zhou, C. Cao, "Weakly Secure Coded Distributed Computing", in *Proc. of the 15th IEEE International Conference on Ubiquitous Intelligence and Computing*, pp. 603-610, 2018.

[9] V. Vu, S. Setty, A. J. Blumberg, M. Walfish, "A Hybrid Architecture for Interactive Verifiable Computation", in *Proc. of the 34th IEEE Symposium on Security and Privacy*, pp. 223-237, 2013.

[10] X. Yu, Z. Yan, A. V. Vasilakos, "A Survey of Verifiable Computation", *Mobile Networks and Applications*, vol. 22, no. 3, pp. 438-453, 2017.

[11] Y. Zhang, Z. Zheng, M. Lyu, "BFTCloud: A Byzantine Fault Tolerance Framework for Voluntary-Resource Cloud Computing", in *Proc. of the 4th IEEE International Conference on Cloud Computing*, pp. 444-451, 2011.

[12] P. L. Aublin, S. B. Mokhtar, V. Quema, "RBFT: Redundant Byzantine Fault Tolerance", in *Proc. of the 33rd IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 297-306, 2013.

[13] R. Garcia, R. Rodrigues, N. Preguica, "Efficient Middleware for Byzantine Fault Tolerant Database Replication", in *Proc. of the 6th European Conference on Computer Systems (EuroSys)*, pp. 107-122, 2011.

[14] R. Gennaro, G. Craig, P. Bryan, "Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers", in *Proc. of the 30th Annual Cryptology Conference*, pp. 465-482, 2010.

[15] W. Song, B. Wang, Q. Wang, C. Shi, W. Lou, Z. Peng, "Publicly Verifiable Computation of Polynomials over Outsourced Data with Multiple Sources", *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 12, no. 10, pp. 2334-2347, 2017.

[16] S. Halevi, V. Shoup, "Faster Homomorphic Linear Transformations in HElib", in *Proc. of Annual International Cryptology Conference. Springer, Cham*, pp. 93-120, 2018.

[17] Q. Yu, M. A. Maddah-Ali, A. S. Avestimehr, "Polynomial Codes: An Optimal Design for High-Dimensional Coded Matrix Multiplication", in *Proc. of the 31st Neural Information Processing Systems (NIPS)*, pp. 4403-4413, 2017.

[18] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A Scalable Framework for Wireless Distributed Computing", *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2643-2654, 2017.

[19] M. Kiamari, C. Wang, A. S. Avestimehr, "On Heterogeneous Coded Distributed Computing", in *Proc. of IEEE Global Communications Conference*, pp. 1-7, 2017.

[20] M. Fahim, H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, P. Grover, "On the optimal recovery threshold of coded matrix multiplication", in *Proc. of IEEE Conference on Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1264-1270, 2017.

[21] Q. Yu, S. Li, M. A. Maddah-Ali, A. S. Avestimehr, "How to optimally allocate resources for coded distributed computing?", in *Proc. of IEEE International Conference on Communications (ICC)*, pp. 1-7, 2017.

[22] Y. Yang, M. Interlandi, P. Grover, S. Kar, S. Amizadeh, and M. Weimer, "Coded Elastic Computing", in *Proc. of 2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 2654-2658, 2019.

[23] L. Tang, K. Konstantinidis, A. Ramamoorthy, "Erasure Coding for Distributed Matrix Multiplication for Matrices With Bounded Entries", *IEEE Communication Letters*, vol. 23, no. 1, pp. 8-11, 2019.

[24] S. Halevi, V. Shoup, "Algorithms in HElib", in *Proc. of the 34th Annual Cryptology Conference*, pp. 554-571, 2014.