Verifying Outsourced Inner Product Computation via Vector Aggregation in Cloud Computing

Qin ZHANG1,∗, Yan XIONG2, Qiwei LU2

1School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
2School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China

Abstract

Outsourced computation has been extensively studied in a number of computer application fields with the increasing prevalence of cloud computing. As the basis and the fundamental operation, inner product computation of vectors, has also been put forwarded. As the service provider of the cloud computing may not be trustworthy, the correctness of the computation results can be corrupted if the service provider is with random fault or not honest (e.g., lazy, malicious, etc). Therefore, it is necessary for the client to verify the correctness of inner product of vectors. Existing verification mechanisms are almost built on the Merkle tree structure and signature chain mechanism, which lack efficiency on both storage and computation aspects. The existing aggregate verification scheme based on algebraic properties is able to ease such problem, however, there exist some serious design defects and lack important details. In order to solve such problems, in this paper we propose the sound inner product verification schemes based on the algebraic properties. Specifically, we first propose the simple aggravate framework, then we analyze its serious defects under smart cloud attack. Furthermore, we design the local aggregate and random aggregate framework to solve the problems respectively. Finally, we provide analysis of the proposed verification schemes in this paper.

Keywords: Inner Product; Outsourced Verification; Vector Aggregation

1 Introduction

Outsourcing computation into the cloud is becoming increasingly popular from two aspects. One is the limitation of the storage and computation resources for data owner, the other is the benefit of rapid development of cloud computing. In such a scenario, massive data and time-consuming task can be completed by the remote cloud service provider(CSP) in a transparent way to data owners. As the basis and fundamental operation, such as similarity measure, document retrieval [7] and components of data mining algorithms (SVM [3], k-means, etc), the outsourcing inner product computation of vectors has been put forwarded and studied significantly.

∗Corresponding author.
Email address: zhangqin@njupt.edu.cn (Qin ZHANG).
Due to the existence of untrusted CSP, privacy and correctness problems is becoming more and more outstanding. To our best of knowledge, much research has been done on the problem of privacy preserving outsourcing inner product computation [2, 10, 11, 12], and the work has been done well.

As for the verification mechanism for the correctness threats, though there exist some related research from database fields which can be summarized as: verification Merkle tree or signature based [4, 5, 8], and the redundant data addition based [1, 13]. They are all general method which doesn’t consider the property of the details of the task. Thus they are not practical enough though effective, especially when the inner product happens on high dimension or large scale data. It is noted that the special schemes combined with the task properties can improve the efficiency of verification well, such inspiration can be seen from associate rule verification [11] and linear algebra verification [6, 10]. Based on the efficient properties of algebra computation, [9] proposed an efficient verification scheme based on the aggregation algebra mechanism, the data owner undertake the responsibility to aggregate the vectors into a the new aggregate vector by some private key, then outsource them into the cloud, the client can verify the inner product correctness with the private key. [9] claimed the soundness of such aggregate scheme with some analysis, but the key details on the encryption of aggregate vector and the verification process is lack some details, thus computational cost isn’t reasonable and sufficient. In fact, the lack of such details will lead the scheme to suffer from severe smart cloud attack and can’t be applicable.

In order to solve such problems, in this paper, we propose sound inner product verification schemes based on the algebraic properties. Specifically, we first propose the simple aggravate framework, then we analyze its serious defects under smart cloud attack. Furthermore, we design the local aggregate and random aggregate framework to solve the problems respectively. Finally, we provide analysis of the proposed verification schemes in this paper.

The rest of this paper is organized as follows. Section 2 presents model descriptions and assumptions. Section 3 shows simple aggregate framework and discuss the security problems under smart cloud attack. Then we further propose two enhanced frameworks in Section 4 and 5, named local and random aggregate framework. In Section 6, we provide the security and performance analysis on the frameworks proposed in this paper. Section 7 concludes the paper.

2 Model and Assumpations

2.1 System model

The cloud system model has been discussed and presents in many articles [10, 12], etc. Here we briefly illustrate the model involving three different entities as follows:

- **Data owner** $O$. Each owner owns vector dataset $D = \{d_i\}_{i \in \{1,...,n\}}$ to outsource into the cloud, where $n$ is the size of $D$, $d_i$s are vectors with $m$ dimensions.

- **Data user or client** $U$. The entity requires the the inner product computation results towards the data stored in the cloud, where the computation is acted according to the request vector $r$ with $m$ dimensions.

- **Cloud service provider as** $C$. The cloud server compute and return the inner product results when data user/client requests.
2.2 Threat model

In this paper, we focus on the verification mechanism and assume some privacy preserving outsourcing mechanism has been adapted [2, 12], thus below we doesn’t consider the traditional semi-honest cloud attack into our threat model in this paper. Now we define the soundness of our verification schemes against the two different cloud threat model as follows:

- **Random fault attack**: there exist some defects on both software, hardware or communication channels, which result in the wrong computation inner product results.

- **Stupid cloud attack**: a stupid cloud service provider who is unaware of the verification mechanism and may be lazy or malicious.

- **Smart cloud attack**: a smart cloud service provider who is knowledge of the verification mechanism well, may be lazy or malicious and do best to make the fake results pass the verification.

3 Simple Aggregation Framework

We have mentioned before that [9] proposed an efficient verification scheme based on the aggregation algebra mechanism, and solve the existing inefficiency problem. However, we find the scheme lose some key details and will result in some significant security threats. Now based on the idea in [9], we start from the simple framework below. The key procedures of the aggregation scheme include **aggregate vector construction** and **client verification**. The details are as follows.

- **aggregate vector construction.** data owner generate the private random vector \( x = [x_1, x_2, \ldots, x_n] \), then construct the aggregate vector \( d_{n+1} \) with the existing vectors \( \{d_i\}_{i \in \{1, \ldots, n\}} \in D \) as \( d_{n+1} = \sum_{i=1}^{n} x_i d_i \), then outsources \((d_1, d_2, \ldots, d_n; d_{n+1}) \) to cloud. The time complexity is \( O(mn) \);

- **cloud computation and return.** cloud server compute the inner product with the request vector \( r \) towards the \((d_1, d_2, \ldots, d_n; d_{n+1}) \) as \( k_i = r \cdot d_i, i = 1, \ldots, n+1 \) and return the results as \((k_1, k_2, \ldots, k_{n+1}) \) to client. Cloud extra time and space complexity is both \( O(m) \);

- **client verification.** authorized client first gets the private random vector \( x = [x_1, x_2, \ldots, x_n] \) from data owner, then compute the \( k'_{n+1} = \sum_{i=1}^{n} x_i k_i \), then compares \( k'_{n+1} = k_{n+1} \), pass the verification only when the equation holds. The time complexity is \( O(n) \);

It is worth mentioning that \((d_1, d_2, \ldots, d_n; d_{n+1}) \) should be protected by some privacy preserving mechanism, e.g., matrix transform or homomorphic in case of the semi-honest cloud. To express clearly, here we neglect them. Now we do some analysis on the simple framework above, it is obvious that it can detect the random fault attack and stupid cloud attack due to the aggregation mechanism utilized above. However, we can find that it will suffer from the smart attack based on the corruption of the private random vector \( x = [x_1, x_2, \ldots, x_n] \).
The smart cloud can establish the equation with the outsourced data:

$$d_{n+1} = \sum_{i=1}^{n} x_i d_i = x_1 d_1 + x_2 d_2 + \cdots + x_n d_n$$

$$\Rightarrow \begin{pmatrix} 
  d_{n+1,1} \\
  \vdots \\
  d_{n+1,m}
\end{pmatrix} = x_1 \begin{pmatrix} 
  d_{11} \\
  \vdots \\
  d_{1m}
\end{pmatrix} + x_2 \begin{pmatrix} 
  d_{21} \\
  \vdots \\
  d_{2m}
\end{pmatrix} + \cdots + x_n \begin{pmatrix} 
  d_{n1} \\
  \vdots \\
  d_{nm}
\end{pmatrix}$$

which can turn into the form as equation set:

$$\begin{align*}
  x_1 d_{11} + x_2 d_{21} + \cdots + x_n d_{n1} - d_{n+1,1} &= 0 \\
  x_1 d_{12} + x_2 d_{22} + \cdots + x_n d_{n2} - d_{n+1,2} &= 0 \\
  \vdots & \vdots \\
  x_1 d_{1m} + x_2 d_{2m} + \cdots + x_n d_{nm} - d_{n+1,m} &= 0
\end{align*}$$

We note that once $m \geq n$, i.e., vector dimension exceeds dataset size(vector number), the equation set can be solved definitely, solution $[x_1, x_2, \ldots, x_n, -1]$ can be easily got, and the place -1 corresponds to the aggregate vector. Thus the smart cloud can create the fake return values $(k_1, k_2, \ldots, k_{n+1})$ satisfying $k_{n+1} = \sum_{i=1}^{n} x_i k_i$ easily. In fact, the such high dimensional data is popular in some specific research fields such as biology, so we can not promise anything. What’s more, if we permit some data update mechanism in the framework (assume on vector $d_1$),

$$d_{n+1}' = x_1 d_1' + x_2 d_2' + \cdots + x_n d_n'$$

we can find that private random vector $x = [x_1, x_2, \ldots, x_n]$ can bring in more than existing $m$ valid equations. In fact, though we change the aggregate constant from -1 to some random parameter $x_{n+1}$, there will be no essential difference when solving Eq. (4).

Here we can have two intuitive solution:

(1) Cancel the uploading of the aggregate vector $d_{n+1}$, so that the smart cloud is not able to establish any equation obviously.

(2) Hide the shape of aggregate vector $d_{n+1}$ uploaded, and change such shape at times.

The first way forbid the establishment of the equation to avoid the break of the $x$, where the owner should keep and transmit the aggregate vector to the client well. The second way present the corruption of $x$ maintaining such uploading mechanism, thus need some extra design. We will explore and analyze the frameworks in the sections below.

4 Local Aggregation Framework

In the local aggregation framework, the data owner keeps the aggregate vector locally. When the client needs such verification, the data owner transmit it. The details can be seen as follows.

- **Aggregate vector construction.** Data owner generate the private random vector $x$, then construct the aggregate vector $d_{n+1}$ with the existing vectors $\{d_i\}_{i=1}^{n} \text{ as } d_{n+1} = \sum_{i=1}^{n} x_i d_i$, then outsources $(d_1, d_2, \ldots, d_n)$ to cloud, stores $d_{n+1}$ locally. The time complexity is $O(mn)$;
Cloud computation and return. Cloud server compute the inner product with the request vector \( r \) towards the \((d_1, d_2, \ldots, d_n)\) as \( k_i = r \cdot d_i, i = 1, \ldots, n \) and return the results as \((k_1, k_2, \ldots, k_n)\) to client. There is no extra storage and computation cost;

Client verification. Authorized client gets the private random vector \( x = [x_1, x_2, \ldots, x_n] \) and aggregate vector \( d_{n+1} \) from data owner, then (1) compute \( k_{n+1} = r \cdot d_{n+1} \); (2) compute the \( k'_{n+1} = \sum_{i=1}^{n} x_i k_i \) from the returned results; (3) compares \( k'_{n+1} \equiv k_{n+1} \), pass the verification only when the equation holds. The time complexity is \( O(m + n) \);

The local framework can support the outsourced data update. That is to say, once data owner request for data update on \( d_l \to d'_l(1 \leq l \leq n) \), the cloud perform such update. Then data owner should update the local aggregate as: \( d_{n+1} \leftarrow d_{n+1} + x_l(d'_l - d_l) \).

## 5 Random Aggregate Framework

Now we don’t utilize the local aggregate framework above, in other words, we will try to hide the shape of aggregate vector \( d_{n+1} \) uploaded, and change such shape at times. The key idea of this framework is to change the shape (add some randomness \( \phi \)) to prevent the establishment of equation. The details can be seen as follows,

\[
d_{n+1} = \sum_{i=1}^{n} x_i d_i = x_1 d_1 + x_2 d_2 + \cdots + x_n d_n + \phi
\]

\[
\Rightarrow \begin{pmatrix} d_{n+1,1} \\ \vdots \\ d_{n+1,m} \end{pmatrix} = x_1 \begin{pmatrix} d_{11} \\ \vdots \\ d_{1m} \end{pmatrix} + x_2 \begin{pmatrix} d_{21} \\ \vdots \\ d_{2m} \end{pmatrix} + \cdots + x_n \begin{pmatrix} d_{n1} \\ \vdots \\ d_{nm} \end{pmatrix} + \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_m \end{pmatrix} \tag{3}
\]

Then the new equation set

\[
\begin{align*}
x_1 d_{11} + x_2 d_{21} + \cdots + x_n d_{n1} - d_{n+1,1} + \phi_1 &= 0 \\
x_1 d_{12} + x_2 d_{22} + \cdots + x_n d_{n2} - d_{n+1,2} + \phi_2 &= 0 \\
\cdots \cdots \\
x_1 d_{1m} + x_2 d_{2m} + \cdots + x_n d_{nm} - d_{n+1,m} + \phi_m &= 0
\end{align*}
\tag{4}
\]

will have numerous solution due to the number of the variables \( m + n + 1 > m \) for solution \([x_1, x_2, \ldots, x_n, -1, \phi_1, \ldots, \phi_m]\). The details are as follows.

- **Aggregate vector construction.** Data owner generate the private random vector \( x \) and \( \phi \), then construct the aggregate vector \( d_{n+1} \) with the existing vectors \( \{d_i\}_{i \in [1, \ldots, n]} \) as \( d_{n+1} = \sum_{i=1}^{n} x_i d_i + \phi \), then outsources \((d_1, d_2, \ldots, d_n; d_{n+1})\) to cloud. The time complexity is \( O(mn) \);

- **Cloud computation and return.** Client send the request vector \( r \), then cloud server compute the inner product towards the \((d_1, d_2, \ldots, d_n; d_{n+1})\) as \( k_i = r \cdot d_i, i = 1, \ldots, n + 1 \) and return the results as \((k_1, k_2, \ldots, k_{n+1})\) to client. Cloud extra time and space complexity is both \( O(m) \);
• **Client verification.** Authorized client first gets the private random vector \( x = [x_1, x_2, \ldots, x_n] \) and \( \phi \) from data owner, then compute the \( k'_{n+1} = \sum_{i=1}^{n} x_i k_i + \phi \), then compares \( k'_{n+1} \equiv k_{n+1} \), pass the verification only when the equation holds. The time complexity is \( O(n) \);

This framework can also support the outsourced data update. Similarly, if data owner requests for data update on \( d_l \rightarrow d'_l (1 \leq l \leq n) \), in case of the equation number addition, he should also update \( \phi \rightarrow \phi' \), which will lead to the update of aggregate vector. That is to say, the cloud should perform the update:

\[
\begin{align*}
    d_l & \rightarrow d'_l \\
    d_{n+1} & \leftarrow d_{n+1} + x_l (d'_l - d_l) + (\phi' - \phi)
\end{align*}
\]  
(5)

In fact, we can regard the random aggregate mechanism here as one special type of encryption whose key is \( \phi \). That is to say, before outsourcing the data, the data owner encrypt the aggregate vector as:

\[
    D' = \{ D; d_{n+1} \} = \{ d_1, d_2, \ldots, d_n; d_{n+1} \} = \{ d_1, d_2, \ldots, d_n; Enc(\phi, d_1, d_2, \ldots, d_n) \}
\]  
(6)

When the client performs the verification, it use the key \( \phi \) to decrypt the aggregate vector. The random key \( \phi \) is unknown to the cloud, which is the crux of the security. It is worth mentioning that [9] mentioned such encryption mechanism, but without details about such encryption, lacking the decryption cost analysis, and the key transmission step. What’s more, if utilizing the DES or RSA encryption mechanism as in [9], it will bring in high computation cost and not applicable.

### 6 Analysis

#### 6.1 Security analysis

In this subsection, we will focus on the security of the frameworks proposed in this paper. Below we will refer the frameworks as **Simple aggregate**, **Local aggregate**, **Random aggregate** respectively. And the summary will be done from three attack aspects, Random fault, Stupid cloud and Smart cloud attack.

First we address that all the three aggregate frameworks can resist Random fault and Stupid cloud attack. This can be deduced from the property of Random fault and Stupid cloud attack, combining with the probabilistic insurance by the aggregation equation. That is to say, in the case of random fault and stupid cloud attack, the probability passing the verification detection is the probability of the aggregation establishment \( d_{n+1} = \sum_{i=1}^{n} x_i d_i \) or \( d_{n+1} = \sum_{i=1}^{n} x_i d_i + \phi \). Without loss of generality, assume the values of \( x_i \) be integers within \([0, l]\), then the probability of succeeding in tampering with \( s (s > 2) \) elements of \( k_i = r \cdot d_i \) is smaller than \( 2/\ell^s \) and bigger than \( 1/\ell^{2s} \) according to [9]. In our case, the \( l \) is large, thus the verification mechanism is effective for Random fault and Stupid cloud attack.

Now we address the case of Smart cloud attack. According to the analysis in Section 3, we can see that the smart cloud can pass the detection by solving the equation set easily. Thus the simple framework can not resist smart cloud attack. As for the other enhanced frameworks, the equation set can’t be hold for Local aggregate case. The random aggregate mechanism also
Table 1: Security comparisons between different frameworks

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Simple aggregate</th>
<th>Local aggregate</th>
<th>Random aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random fault attack</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Stupid cloud attack</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Smart cloud attack</td>
<td>-</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Table 2: Performances and costs of different frameworks

<table>
<thead>
<tr>
<th>Party</th>
<th>Extra cost</th>
<th>Simple aggregate</th>
<th>Local aggregate</th>
<th>Random aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>Time</td>
<td>$O(mn)$</td>
<td>$O(mn)$</td>
<td>$O(mn)$</td>
</tr>
<tr>
<td></td>
<td>Space</td>
<td>$O(n)$</td>
<td>$O(m + n)$</td>
<td>$O(m + n)$</td>
</tr>
<tr>
<td>Cloud</td>
<td>Time</td>
<td>$O(m)$</td>
<td>-</td>
<td>$O(m)$</td>
</tr>
<tr>
<td></td>
<td>Space</td>
<td>$O(m)$</td>
<td>-</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>Client</td>
<td>Time</td>
<td>$O(n)$</td>
<td>$O(m + n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>Space</td>
<td>$O(n)$</td>
<td>$O(m + n)$</td>
<td>$O(m + n)$</td>
</tr>
</tbody>
</table>

maintain the difficulty in solving the linear equation set with the property that number of the variables $m + n + 1 > m$ exceeds the equation number. Therefore, it is effective with smart cloud attack. The summary can be seen in Table 1 for concision.

6.2 Efficiency analysis

As for the efficiency analysis, we do the analysis for the three frameworks from two aspects, i.e., extra time cost and space cost result from the verification mechanism. We summarize them in Table 2. The extra time costs for data owner are all $O(mn)$ due to the extra aggregation. The space cost for data owner is different. The simple aggregate framework only occupies $O(n)$ for the storage of the privacy vector $[x_1, x_2, \ldots, x_n]$. The other frameworks need the extra $O(m)$ space cost for the local storage of $d_{n+1}$ and random vector $\phi$ respectively. The space cost is the same.

The extra time costs for data owner are all $O(mn)$ due to the extra aggregation. The space cost for data owner is different. The simple aggregate framework only occupies $O(n)$ for the storage of the privacy vector $[x_1, x_2, \ldots, x_n]$. The other frameworks need the extra $O(m)$ space cost for the local storage of $d_{n+1}$ and random vector $\phi$ respectively. The space cost is the same.

Except the local aggregate framework need the extra computation of inner product on unloaded $d_{n+1}$ with cost $O(m)$. The other two are both $O(n)$ without any inner product computation. The other two both need the extra inner product computation with $O(m)$ cost. Considering the space cost, the simple aggregate framework only needs extra storage of the private vector $[x_1, x_2, \ldots, x_n]$ with $O(n)$ (the returned results from cloud is not the extra by verification). The local framework occupies $O(m)$ more cost for $d_{n+1}$. And the random aggregate framework also needs $O(m)$ for the random vector $\phi$ compared with simple aggregate case.

We can see that the random shift $O(m)$ computation cost from client to cloud and thus need $O(m)$ more cloud space than local aggregate framework.
7 Conclusion

The existing verification mechanisms are almost built on the Merkle tree structure and signature chain mechanism, which lack efficiency on both storage and computation aspects. The existing aggregate verification scheme based on algebraic properties is able to ease such problem, however, there exist some serious design defects and lack important details. In order to solve such problems, in this paper we propose the sound inner product verification schemes based on the algebraic properties. Specifically, we first propose the simple aggravate framework, then we analyze its serious defects under smart cloud attack. Furthermore, we design the local aggregate and random aggregate framework to solve the problems respectively. Finally, we provide analysis of the proposed verification schemes in this paper.

References