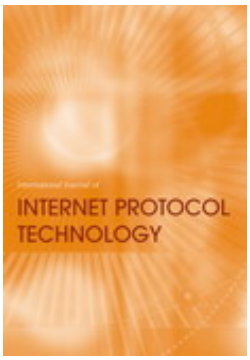
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Internet Protocol Technology

ISSN online: 1743-8217 - ISSN print: 1743-8209
https://www.inderscience.com/ijipt

Secure and verifiable outsourcing of Euclidean distance and closest pair of points with single untrusted cloud server

Shilpee Prasad, B.R. Purushothama

DOI: 10.1504/IJIPT.2023.10054905

Article History:

Received:	24 May 2020
Last revised:	23 February 2021
Accepted:	03 June 2021
Published online:	23 March 2023

Secure and verifiable outsourcing of Euclidean distance and closest pair of points with single untrusted cloud server

Shilpee Prasad* and
B.R. Purushothama

Department of Computer Science and Engineering,
National Institute of Technology Goa,
Farmagudi, Ponda, Goa 403401, India
Email: shilpeeprasad34@gmail.com
Email: puru@nitgoa.ac.in
*Corresponding author

Abstract: Owing to resource constraints, often, a client has to outsource the computation to the untrusted cloud service provider. As cloud service providers are often untrusted, the computation's result should be verified for correctness. Also, the cost of verification should be less than the cost of actual computation. In this paper, we address the problem of verifying the computation of a geometric problem. In particular, we address the problem of verifying the Euclidean distance and closest pair of points returned by the single untrusted cloud service provider. We have designed verification schemes for outsourcing Euclidean distance and the closest pair of points. We have proved that the proposed scheme has negligible server cheating probability. Also, the scheme preserves the privacy of the outsourced data. We have implemented the closest pair of points verification scheme and show that the verification cost is significantly less than the actual computation cost. Also, compared to the existing scheme, the proposed scheme has less server cheating probability.

Keywords: outsource computation; closest pair of points; Euclidean distance; privacy preserving; cloud service provider.

Reference to this paper should be made as follows: Prasad, S. and Purushothama, B.R. (2023) 'Secure and verifiable outsourcing of Euclidean distance and closest pair of points with single untrusted cloud server', *Int. J. Internet Protocol Technology*, Vol. 16, No. 1, pp.34–45.

Biographical notes: Shilpee Prasad is a post-graduate student at the Department of Computer Science and Engineering, National Institute of Technology Goa. She works in the area of Information Security and Cryptography, Key management and Security Analytics.

B.R. Purushothama obtained his PhD in Computer Science and Engineering from National Institute of Technology Warangal (NIT Warangal), India and did his MTech in Computer Science and Engineering from National Institute of Technology Karnataka (NITK) Surathkal, India. He is currently working as Associate Professor in the Department of Computer Science and Engineering at National Institute of Technology Goa, India. His areas of interest are cryptography, key management, cloud security, data security, provable security, network security and algorithms.

1 Introduction

Cloud computing offers storage and computation services for the users, thereby eliminating the need for purchasing the costly storage and computing resources by the users. The users often outsource their data and computation to the cloud service provider (CSP). The cloud service provider is often honest-but-curious or untrusted. The data owner should ensure the confidentiality of data with a piece of sensitive information by encrypting the data and outsourcing it to the CSP. Not only the data which could be outsourced to the cloud but also computations. The resource-constrained devices can outsource computational intensive tasks to CSP and use unlimited computing resources by paying as per the usage. However, the CSP is not trusted to return the correct result of the computation. Thus, there is a necessity to verify the computation result by the user/client who has outsourced the computation. The challenge is that the computation cost to verify the correctness of the result returned by the CSP should be less than the computation cost of computing the result. For instance, consider two matrices P and Q of order n. Suppose a user outsources the task of computing the product of two matrices to CSP. Let the result returned by the CSP be the matrix R of order n. The result returned by the CSP might

be incorrect as CSP is untrusted. So, the user should verify whether the result computed by the CSP is correct or not. The cost of computing the product of the two n order matrices is $O(n^3)$. However, the cost of verifying the result should be less than $O(n^3)$. The motivation for the CSP not to compute the correct result may be to save the computations. So, the user should detect a "cheating" CSP. Any outsourcing computation should ensure that the probability that a CSP cheats the user should be negligible. There are works which addresses verification of the correctness of the outsourced computations such as modular exponentiation (Zhu et al., 2018), polynomial function evaluation (Li et al., 2013), vector inner product (Sheng et al., 2013), matrix multiplication (Benjamin and Atallah, 2008) etc. In this paper, we focus on verifying the correctness of a geometric problem.

1.1 Our contribution

Our contributions are summarised below.

- We propose a privacy-preserving verification scheme to verify the correctness of Euclidean distance between any two given points.
- We propose a privacy-preserving verification scheme to verify the correctness of the closest pair of points among the given n points.
- We analyse the proposed verification schemes and show that the probability of CSP cheating is negligible.
- We analyse the performance of the proposed verification scheme for the Closest pair of points and show that the computation cost of verification is too low compared to the CSP computation cost.

1.2 Organisation

The rest of the paper is organised as below. Related work is given in Section 2. Section 3 gives the system model and preliminaries. The proposed scheme for verifying the Euclidean distance and closest pair of points is given in Section 4 and Section 5 respectively. Section 6 provides performance analysis and comparison, followed by the conclusion in Section 7.

2 Related work

Due to resource constraints on the clients, the computations are often outsourced to the untrusted cloud service provider. However, the result of the computation needs to be verified as the CSP might not do the legal computations and might send arbitrary random results. So, it is crucial to verify the result returned by the CSP. This paper aims to verify the correctness of the Euclidean distance and the closest pair of points. There are works wherein some of the critical problems have been addressed in this model of verification. Also, preserving the privacy of the data is important. Here, we highlight some of the problems for which the verification schemes have been designed. Modular exponentiation is one of the potential problems to be outsourced since it is used in most of the cryptographic algorithms. Ye et al. (2016) proposed an algorithm for outsourcing modular exponentiation using a two-server model, and for preserving the privacy of input data, a mathematical division operation is used. Since the algorithm is based on two-server model, it suffers from the collusion attack. Shuai et al. (2017) proposed an algorithm for outsourcing composite modular exponentiation in a single server model.

Benjamin and Atallah (2008) introduced an algorithm for outsourcing matrix multiplication using a two-server model, and verification is done by performing vector multiplication. This scheme, however, suffers from collusion attack due to the two-server model. To solve the collusion problem of Benjamin and Atallah (2008) scheme, Atallah and Frikken (2010) proposed an improved protocol for matrix multiplication using a single server model. Zhang et al. (2016b) proposed an algorithm for matrix multiplication using public verification. In this scheme, matrices are sent to the CSP in the offline mode, and hence scheme is not suitable for real-time applications. Erfan and Mal (2020) constructed matrix multiplication scheme which is publicly verifiable and works in online mode, hence can be used for real-time applications. Lei et al. (2013) proposed an algorithm for outsourcing large matrix inversion; this scheme addresses the privacy of input data only for non zero elements in the matrix. Zhang et al. (2016a) proposed a protocol for outsourcing the system of linear matrix equations in which the permutation technique is used to encrypt the linear matrix equations. Sheng et al. (2013) proposed an algorithm for the outsourcing of inner product of the vectors for verification they are using the concept of aggregate vector; however, this scheme is not publicly verifiable.

Sion (2005) proposed a method for query execution assurance for outsourced databases using verification schemes based on single challenge token, multiple challenge token and fake challenge token. This scheme focuses on read-only queries; update queries are processed with additional cost. Le and Li (2012) proposed query access assurance for IO-bound queries in the distributed databases. Xue et al. (2018) designed a verifiable outsourcing identity-based encryption scheme for private key generation. Li et al. (2019) designed verifiable outsourced encryption and decryption scheme using attribute-based encryption. Xixun et al. (2019) designed a publicly verifiable scheme for encrypted data; this scheme uses fully homomorphic encryption hence not efficient.

The closest pair of points has many applications such as computational biology, geographic information system, computational finance, weather prediction. Kuruba et al. (2016) proposed an algorithm for outsourcing the closest pair of points for verification using the concept of single challenge token proposed by Sion (2005). Kuruba et al. (2016) method uses two-server model and homomorphic encryption (Murugesan et al., 2010) for data privacy. Hence

Kuruba et al. (2016) scheme suffers from collusion attack, also due to the use of homomorphic encryption, it is not efficient. Our proposed model is based on a single server model and does not use homomorphic encryption, and the probability of server cheating is negligible.

3 System model and preliminaries

The system model includes two entities, Client C and cloud service provider S. Cloud service provider S is untrusted. Client C outsources a computational task to S. Cloud server performs the computational task given by client C and returns the result to C. Then, client C verifies the result returned by S for its correctness. Note that the cloud service provider and cloud server are used interchangeably.

We propose two schemes; one for verifying the correctness of the Euclidean distance computation and the other for verifying the correctness of the closest pair of points.

The proposed scheme Π_1 is defined to have three phases.

$\Pi_1 = \{\text{Outsource, Response Computation, Verification}\}$.

1. **Outsource**: C sends the two points $\{P_1, P_2\}$ to S.
2. **Response Computation**: S computes the Euclidean distance d of $\{P_1, P_2\}$ and sends d to C.
3. **Verification**: C verifies whether d is correct.

The proposed scheme Π_2 is defined to have three phases.

$\Pi_2 = \{\text{Outsource, Response Computation, Verification }\}$.

1. **Outsource**: C sends the set of points $P = \{P_1, \ldots, P_n\}$ to S.
2. **Response Computation**: S computes the distance d and sends d to C.
3. **Verification**: C verifies whether d is the minimum distance among the points in P.

3.1 Notations

- Point $P_i = (x_i, y_i)$, where x_i and y_i are values from two-coordinate system.
- $Euclidean\_Distance(P_i, P_j)$ is defined as $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$

4 Proposed scheme for verifying the correctness of Euclidean distance

In this section, we elaborate the scheme to verify the correctness of Euclidean distance of two points computed by the CSP. The proposed scheme comprises of three phases; Client Computation, CSP Computation and Client Verification.

4.1 Outsourcing of computation by client

The process of Client outsourcing the computation to S is elaborated in Algorithm 1.

Algorithm 1: Client computation for outsourcing

Input: Points $P_i = (x_i, y_i), P_j = (x_j, y_j)$

Output: $(P_i^{(u)}, P_j^{(u)})$, $(P_i^{(v)}, P_j^{(v)})$ to be sent to the CSP

1 Client chooses randomly u and computes:

2 $P_i^{(u)} = (x_i + u, y_i + u)$

3 $P_j^{(u)} = (x_j + u, y_j + u)$

4 Client chooses randomly v and m and computes:

5 $P_i^{(v)} = (m(x_i + v), m(y_i + v))$

6 $P_j^{(v)} = (m(x_j + v), m(y_j + v))$

7 Client sends $(P_i^{(u)}, P_j^{(u)})$, $(P_i^{(v)}, P_j^{(v)})$ to CSP.

Let $P_i = (x_i, y_i)$ and $P_j = (x_j, y_j)$ be the two points. Client C chooses randomly u and computes points $P_i^{(u)} = (x_i + u, y_i + u)$, $P_j^{(u)} = (x_j + u, y_j + u)$. This is carried out so as to not to reveal the original points to S. Also, C chooses randomly v and m and computes $P_i^{(v)} = (m(x_i + v), m(y_i + v))$, $P_j^{(v)} = (m(x_j + v), m(y_j + v))$. Note that the points $P_i^{(v)}$ and $P_j^{(v)}$ are computed from the original points P_i and P_j. The random v is used to hide the points and m is used during the verification. C sends $\{(P_i^{(u)}, P_j^{(u)}), (P_i^{(v)}, P_j^{(v)})\}$ to the CSP. C keeps u, v and m as secret.

4.2 Response computation by CSP

CSP after receiving $\{(P_i^{(u)}, P_j^{(u)}), (P_i^{(v)}, P_j^{(v)})\}$ from the client C, computes the Euclidean distance between these pair of points as given in Algorithm 2 and sends them to the C.

Algorithm 2: CSP computation

Input: $(P_i^{(u)}, P_j^{(u)})$, $(P_i^{(v)}, P_j^{(v)})$ sent by the client.

Output: (α, β) to be sent to the client.

1 Server computes following:

2 $\alpha = Euclidean\_distance(P_i^{(u)}, P_j^{(u)})$

3 $\beta = Euclidean\_distance(P_i^{(v)}, P_j^{(v)})$

4 Server sends (α, β) to the Client.

4.3 Client verification

After receiving the response from the CSP, client C verifies whether the distance computed is correct or not by using Algorithm 3.

Algorithm 3: Client verification

Input: (α, β) sent by the server.

Output: Verification of Euclidean distance computed by the server.

1 Client computes $\gamma = m\alpha$

2 $\gamma = \beta$ **then**

3 Print "Correct Euclidean distance computation"

4 **end**

5 **else**

6 Print "Incorrect Euclidean distance computation"

7 **end**

4.3.1 Correctness of verification

In this section, we prove the correctness of the verification procedure as given in **Algorithm 4.3** that is being used by the client for verification. Server S sends (α, β) to the client, where $\alpha = Euclidean\_distance(P_i^{(u)}, P_j^{(u)})$ and $\beta = Euclidean\_distance(P_i^{(v)}, P_j^{(v)})$. For the computation to be correct, $\gamma = \beta$ should satisfy, where $\gamma = m\alpha$. Note that,

$$\begin{aligned}\alpha &= Euclidean\_distance(P_i^{(u)}, P_j^{(u)}) \\ &= \sqrt{((x_j+u)-(x_i+u))^2 + ((y_j+u)-(y_i+u))^2} \\ &= \sqrt{(x_j+u-x_i-u)^2 + (y_j+u-y_i-u)^2} \\ \alpha &= \sqrt{(x_j-x_i)^2 + (y_j-y_i)^2}\end{aligned}$$

and

$$\begin{aligned}\beta &= Euclidean\_distance(P_i^{(v)}, P_j^{(v)}) \\ &= \sqrt{(m(x_j+v)-m(x_i+v))^2 + (m(y_j+v)-m(y_i+v))^2} \\ &= \sqrt{m^2(x_j+v-x_i-v)^2 + m^2(y_j+v-y_i-v)^2} \\ &= m\sqrt{(x_j-x_i)^2 + (y_j-y_i)^2}\end{aligned}$$

So, it can be observed that $\beta = m\alpha$.

4.4 Computational cost

This section analyses the computation cost for verification, computation cost for challenge generation, and computation cost at the CSP. Table 1 gives the details of the computational cost. It can be observed that the cost of verification by the client is less than the cost of server computation. *The saving in the computation at verification becomes very significant when we use this approach in our proposed scheme to verify the closest pair of points.*

Table 1 Computational cost-Euclidean distance

Computation	*#Addition*	*#Subtraction*	*#Multiplication*	*#Square root*
Computational Cost for client verification	0	0	1	0
Computational Cost for Challenge generation	8	0	4	0
Computational Cost at Server	2	4	4	2

5 Proposed scheme for verifying the correctness of closest pair of points

In this section, we elaborate the scheme for verifying the correctness of the computation of the closest pair of points. The closest pair of points problem involves n points and expects to compute the closest pair of points among the given n points. We address the problem of outsourcing the computation of closest pair of points distance to the untrusted CSP and verifying the correctness of the result returned by the CSP by the client.

The proposed scheme consists of three phases; Client computation and outsourcing, CSP Response Computation and Client Verification. In particular, in the first phase, the client outsources the task of computing minimum distance points to the CSP, followed by the second phase wherein the CSP computes the distance and returns it to the client. Finally, in the third phase, the client verifies the correctness of the returned distance.

5.1 Client computation and outsourcing

Suppose that the client has a set of n points and wants the closest pair of points among n points. The detailed algorithm for Client computation and outsourcing is given in Algorithm 4.

Algorithm 4: Client computation and outsourcing

Input: Set of n points, $S_1 = \{P_1, \ldots, P_n\} = \{\{x_1, y_1\}, \ldots, \{x_n, y_n\}\}$

Output: Input set $S_4 = \{P_1, P_2, \ldots, P_{n+r}\}$ consisting of $(n+r)$ points to be sent to the CSP.

1 $S_2 = \{\}, S_3 = \{\}, S_4 = \{\}$

2 Client randomly chooses u

3 **for each** $P_i \in S_1$ **do**

4 $P_i^{(u)} = (x_i + u, y_i + u)$

5 $S_2 = S_2 \cup \{P_i^{(u)}\}$

6 **end**

7 $S_4 = S_4 \cup S_2$

$S_2 = \{P_1^{(u)}, P_2^{(u)}, P_3^{(u)}, \ldots, P_n^{(u)}\} = \{\{x_1 + u, y_1 + u\}, \{x_2 + u, y_2 + u\}, \{x_3 + u, y_3 + u\}, \ldots, \{x_n + u, y_n + u\}\}$

Client adds r number of extra points to set S_4.

8 Choose randomly r points from set S_1.

9 Let $S_1^{'} = \{P_{i_1}, P_{i_2}, P_{i_3}, \ldots, P_{i_r}\}$ be the set of r points selected from S_1.

Note: $\{P_{i_1}, P_{i_2}, P_{i_3}, \ldots, P_{i_r}\} \in \{P_1, P_2, P_3, \ldots, P_n\}$

10 Choose randomly m and v.

11 **for each** $(x_i, y_i) \in S_1^{'}$ **do**

12 $P_i^{(v)} = (m(x_i + v), m(y_i + v))$

13 $S_3 = S_3 \cup \{P_i^{(v)}\}$

14 **end**

15 $S_4 = S_4 \cup S_3$ $\quad |S_4| = n + r$

16 Let $j_1, j_2, j_3, ..., j_r$ be the indices of the points of S_3 present in S_4.

17 Let $k_1, k_2, k_3, ..., k_r$ be the indices of the points of S_2 corresponding to S_3 present in S_4.

$S_4[k_1]$ = Element(point) present in the set S_4 at index k_1. Corresponding points means that if $S_4[k_1] = (x_{k_1} + u, y_{k_1} + u)$ then $S_4[j_1] = (m(x_{j_1} + v), m(y_{j_1} + v))$ where $x_{k_1} = x_{j_1}$ and $y_{k_1} = y_{j_1}$.

18 Client sends S_4 to CSP, and instructs CSP to maintain the order of points in S_4.

To preserve data privacy, client masks the original points by adding a randomly chosen value u to all the points in input set. Let the original set of points be $S_1 = \{P_1, \ldots, P_n\} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. After masking the points, let the new set be $S_2 = \{P_1^{(u)}, \ldots, P_n^{(u)}\} = \{(x_1 + u, y_1 + u), \ldots, (x_n + u, y_n + u)\}$.

Client adds r number of extra points to the set S_4, where $S_4 = \{P_1^{(u)}, \ldots, P_n^{(u)}\} = \{(x_1 + u, y_1 + u), \ldots, (x_n + u, y_n + u)\}$ and r is very small compared to n. To carry out this, client randomly chooses r points from original input set S_1 to form new set $S_1^{'}$ of r points. Client randomly chooses constant value m and v and forms another set S_3, where $S_3 = \{P_1^{(v)}, \ldots, P_r^{(v)}\} = \{\{m(x_1 + v), m(y_1 + v)\}, \ldots, \{m(x_r + v), m(y_r + v)\}\}$. Finally, a new set of points S_4 is formed such that $S_4 = S_3 \cup S_4$. So, $|S_4| = n + r$.

Let $j_1, j_2, \ldots, j_r$ be the indices of the points of S_3 present in S_4 and $k_1, k_2, \ldots, k_r$ be the indices of the points of S_2 corresponding to S_3 present in S_4. Here corresponding point means that the same point (x_i, y_i) from set S_1 has been chosen to form the points $S_4[j]$ and $S_4[k]$ where if $S_4[k] = (x_k + u, y_k + u)$ then $S_4[j] = (m(x_j + v), m(y_j + v))$ where $x_k = x_j$ and $y_k = y_j$. Now, Client sends the set S_4 to the CSP and asks the CSP to maintain the order of points.CSP though untrusted, is assumed to maintain the order of points. This is a reasonable assumption, since the CSP has no motivation to change the order. Changing the order of the points indeed limits the server capability towards result computation than assisting it. Detailed algorithm for client computation is given Algorithm 4.

5.2 Response computation by CSP

The detailed algorithm of response computation by CSP is elaborated in Algorithm 5.

Algorithm 5: CSP computation

Input: Input set $S_4 = \{P_1, P_2, \ldots, P_{n+r}\}$ sent by the client.

Output: Euclidean distance matrix $M_{(n+r)*(n+r)}$, Euclidean distance set $D^{''}$ and corresponding points $\{(M_1, Q_1), (M_2, Q_2), \ldots, (M_d, Q_d)\}$ to be sent to the client.

1 $D = \{\}$

2 $M = []$

3 **for** $i \leftarrow 1$ **to** $(n + r)$ **do**

4 **for** $j \leftarrow i + 1$ **to** $(n + r)$ **do**

5 $D_i = Euclidean\_distance(S_4[i], S_4[j])$

$D = D \cup \{D_i\}$

$M[i, j] = D_i$

$M[j, i] = D_i$

end

end

CSP sends Euclidean distance matrix $M_{(n+r)*(n+r)}$ to the client.

Let $D' = \{d_1, d_2, \ldots, d_{\frac{(n+r)(n+r-1)}{2}}\}$ be the Euclidean distances in D in sorted order.

Let $d = (nr + \frac{r(r-1)}{2})$

Let D'' be the set of first d number of distances in D''.

CSP sends Euclidean distance set D'' and corresponding points $\{(M_1, Q_1), \ldots, (M_d, Q_d)\}$ to the client. Where $M_i = (x_i, y_i)$ and $Q_t = (x_t, y_t)$.

Note: n =Actual number of input points in S_4, r =Extra points added in input set S_4

CSP receives S_4 with $n+r$ points and computes Euclidean distance for all $(n+r)$ points present in input set S_4 and forms the Euclidean distance matrix $M_{(n+r)*(n+r)}$. Let set D contains all ${}^{n+r}C_2$ Euclidean distances. Note that CSP will compute only ${}^{n+r}C_2$ distances as these are the distances possible with $n+r$ points, here C represents combination. Let D' be the set of Euclidean distances present in D in sorted order. Let D'' contain the first d number of distances from D' where $d = (nr + \frac{r(r-1)}{2})$. CSP sends Euclidean distance matrix $M_{(n+r)*(n+r)}$, Euclidean distance set D'' and corresponding points $\{(M_1, Q_1), (M_2, Q_2), \ldots, (M_d, Q_d)\}$ to the client, where $M_i = (x_i, y_i)$ and $Q_t = (x_t, y_t)$. Detailed algorithm for CSP computation is given in Algorithm 5.

5.3 Verification of the result by the client

The detailed algorithm for verifying the result returned by the CSP is given in Algorithm 6.

Algorithm 6: Client verification

Input: Euclidean distance matrix $M_{(n+r)*(n+r)}$, Euclidean distance set D'' and corresponding points $\{(M_1, Q_1), (M_2, Q_2), \ldots, (M_d, Q_d)\}$ sent by the server.

Output: Closest pair of points $M_i(x_{m_i}, y_{m_i})$ and $Q_i(x_{q_i}, y_{q_i})$

for $i \leftarrow 1$ **to** r **do**

  $\alpha = Euclidean\_distance(S_4[k_i], S_4[k_{i+1}])$

  $\beta = Euclidean\_distance(S_4[j_i], S_4[j_{i+1}])$

  if $\beta != m\alpha$ **then**

    print "Incorrect Euclidean distance computation by server"

  end

  else

    $i++;$

    Continue

  end

end

if $i == r$ **then**

  Print "Correct Euclidean distance computation by server"

end

Find closest pair of points

if point is from additional point set S_3 then don't consider that point for closest pair

for $i \leftarrow 1$ **to** d **do**

  if $M_i \in S_3$ or $Q_i \in S_3$ **then**

    $i++$

    Continue

  end

  else

    $M_i = (x_{m_i} - u, y_{m_i} - u)$

    $Q_i = (x_{q_i} - u, y_{q_i} - u)$

    Print "Closest Pair of points (M_i, Q_i)"

  end

end

Client receives Euclidean distance matrix $M_{(n+r)*(n+r)}$, Euclidean distance set D'' and corresponding points $\{(M_1, Q_1), (M_2, Q_2), \ldots, (M_d, Q_d)\}$ from the CSP. First, the client verifies whether the Euclidean distance computation done by the server is correct or not. To verify this, client has indices $\{k_1, k_2, \ldots, k_r\}$ of original r points and corresponding indices of additional points $\{j_1, j_2, \ldots, j_r\}$ in set S_4. Assume $\alpha = Euclidean\_distance(S_4[k_i], S_4[k_{i+1}])$ and $\beta = Euclidean\_distance(S_4[j_i], S_4[j_{i+1}])$. For all r indices, the condition $\beta = m\alpha$ should be satisfied for the computed Euclidean distances to be correct by the CSP. Even if the condition fails for one index, then it means that the server has computed incorrect Euclidean distance. If all the resulting r distances computed by the CSP are correct, then CSP is honest with the probability $(1 - \frac{r!r!(n-r)!}{(n+r)!})$. After Euclidean distance verification and finding that the CSP has computed the correct Euclidean distances, the client will find

the closest pair of points using the set D'', which is having d number of distances. Starting from the first distance client will check whether the corresponding points with the distance belong to the extra added point set S_3. If it is so, the client will ignore that distance and check for the next distance in D'' and corresponding point until a distance for which both the points belong to the original input set is found. The detailed algorithm for Client verification is given in Algorithm 6.

5.4 Correctness of verification procedure

In this section, we prove the proposed verification scheme's correctness for the closest pair of points. First, we establish the correctness of the Euclidean distance computation.

5.4.1 Correctness of Euclidean distance computation

To prove this, we have to show that $\beta = m\alpha$. For convenience, let E_d denote $Euclidean\_distance$.

$$\begin{aligned}\alpha &= Euclidean\_distance(S_4[k_i], S_4[k_{i+1}])\\ &= E\_d((x_{k_i}+u, y_{k_i}+u), (x_{k_{i+1}}+u, y_{k_{i+1}}+u))\\ &= \sqrt{((x_{k_i}+u)-(x_{k_{i+1}}+u))^2+((y_{k_i}+u)-(y_{k_{i+1}}+u))^2}\\ &= \sqrt{(x_{k_i}+u-x_{k_{i+1}}-u)^2+(y_{k_i}+u-y_{k_{i+1}}-u)^2}\\ \alpha &= \sqrt{(x_{k_i}-x_{k_{i+1}})^2+(y_{k_i}-y_{k_{i+1}})^2}\end{aligned}$$

and

$$\begin{aligned}\beta &= Euclidean\_distance(S_4[j_i], S_4[j_{i+1}])\\ &= E\_d(((m(x_{j_i}+v), m(y_{j_i}+v)),\\ &\quad (m(x_{j_{i+1}}+v), m(y_{j_{i+1}}+v)))\\ &= E\_d(((m(x_{k_i}+v), m(y_{k_i}+v)),\\ &\quad (m(x_{k_{i+1}}+v), m(y_{k_{i+1}}+v)))\\ &= \sqrt{\begin{aligned}&(m(x_{k_i}+v)-m(x_{k_{i+1}}+v))^2\\ &+(m(y_{k_i}+v)-m(y_{k_{i+1}}+u))^2\end{aligned}}\\ &= \sqrt{\begin{aligned}&m^2(x_{k_i}+v-x_{k_{i+1}}-v)^2\\ &+m^2(y_{k_i}+v-y_{k_{i+1}}-v)^2\end{aligned}}\\ \beta &= m\sqrt{(x_{k_i}-x_{k_{i+1}})^2+(y_{k_i}-y_{k_{i+1}})^2}\end{aligned}$$

It can be observed that $\beta = m\alpha$. Note that $S_4[k_i]$ and $S_4[j_i]$ are the points which have been formed using the same input point (x_i, y_i) from set S_1, where $S_4[k_i] = (x_i + u, y_i + u)$ and $S_4[j_i] = (m(x_i + v), m(y_i + v))$. Therefore, $x_{j_i} = x_{k_i}$, $y_{j_i} = y_{k_i}$, $x_{j_{i+1}} = x_{k_{i+1}}$ and $y_{j_{i+1}} = y_{k_{i+1}}$.

Next, we prove that the closest pair of point distance should be one among the $(nr + (r(r-1))/2 + 1)$ distances sent by the CSP.

Theorem 1: *The closest pair of point distance should be one among the* $(nr + (r(r-1))/2 + 1)$ *distances sent by the CSP.*

Proof. Proof Let n be actual number of points in input set S_2 and let r be the extra number of points(S_3) added to input set S_4 which also contains the points in S_2. Define,

$N_{12}(S_3)$ = the number of distances containing one or both point from set S_3.

$N_1(S_3)$ =Number of distances containing one point from set S_3.

$N_2(S_3)$ =Number of distances containing both the points from set S_3.

So, $N_{12}(S_3) = N_1(S_3) + N_2(S_3)$. And,

$$N_1(S_3) = \underbrace{n + n + \ldots + n}_{r \text{ times}} = nr.$$

This is because, for each point present in the set S_3 ($|S_3| = r$), Euclidean distance is computed with every point present in the original input set S_2 ($|S_2| = n$).

$N_2(S_3) = {}^{r}C_2 = \dfrac{r!}{2!(r-2)!} = \dfrac{r(r-1)}{2}$. These are the possible total number of distances that can be computed with r points.

And, $N_{12}(S_3) = nr + \dfrac{r(r-1)}{2}$.

At most $(nr + \dfrac{r(r-1)}{2})$ number of distances will contain the points from extra added r point set S_3. Hence $(nr + \dfrac{r(r-1)}{2} + 1)$ number of distances ensures that there will be at least one distance which will have both the points from original input point set S_2. CSP provides first $(nr + \dfrac{r(r-1)}{2} + 1)$ number of sorted distances as a response. Hence, closest pair of point distance will be one among $(nr + \dfrac{r(r-1)}{2} + 1)$ number of distances.

5.5 Server cheating probability

In this section, we compute the server cheating probability. Server cheating probability depends on guessing extra added r points and corresponding original points.

Theorem 2: *The CSP can cheat the client with the probability* $\dfrac{r!r!(n-r)!}{(n+r)!}$.

Proof. Proof Let $P(e,o)$ be the probability of server guessing extra added r points and corresponding original points. Let $P(e)$ be the probability of guessing extra r points and $P(o)$ be the probability of guessing corresponding r original points.

So, $P(e,o) = P(e) * P(o)$

$$P(e) = \frac{r}{(n+r)} \cdot \frac{r-1}{(n+r-1)} \cdots \frac{r-(r-1)}{(n+r-(r-1))}$$

$$= \frac{r}{(n+r)} \cdot \frac{r-1}{(n+r-1)} \cdots \frac{1}{(n+1)}$$

$$= \frac{r!}{\frac{(n+r)!}{n!}}$$

$$P(e) = \frac{r!n!}{(n+r)!}$$

and

$$P(o) = \frac{1}{{}^{n}C_{r}} = \frac{r!(n-r)!}{n!}$$

$$So, P(e,o) = \frac{r!n!}{(n+r)!} \cdot \frac{r!(n-r)!}{n!} = \frac{r!r!(n-r)!}{(n+r)!}$$

CSP chooses first point from the input set S_4 containing $(n+r)$ points. The probability of choosing first point from extra added r points is $\frac{r}{n+r}$. Next CSP chooses second point from remaining $(n+r-1)$ points, the probability that this point is chosen from remaining $r-1$ points is $\frac{(r-1)}{n+r-1}$. CSP continues the selection until it chooses all r points. There are ${}^{n}C_{r}$ combinations possible for selecting r points from n points out of which one combination will match the sequence of points selected in $P(e)$ therefore $P(o) = \frac{1}{{}^{n}C_{r}}$. As we increase the value of r the probability of server cheating decreases and becomes almost negligible.

5.6 Computational cost

Table 2 gives the computation cost of outsource, server computation and verification cost. It can be observed that the cost of verification is $O(r+d)$ whereas the cost of result computation by the CSP is $O(n^2)$. So, the cost of verification is less in comparison with the actual computation.

Table 2 Computational cost-closest pair of points

Computation	*# of operations*	*Complexity*
Outsource of computation(Client)	$2r+2r$	$O(r)$
Server computation	$n^2 + nlog(n)$	$O(n^2)$
Verification(Client)	$r+d$	$O(r+d)$

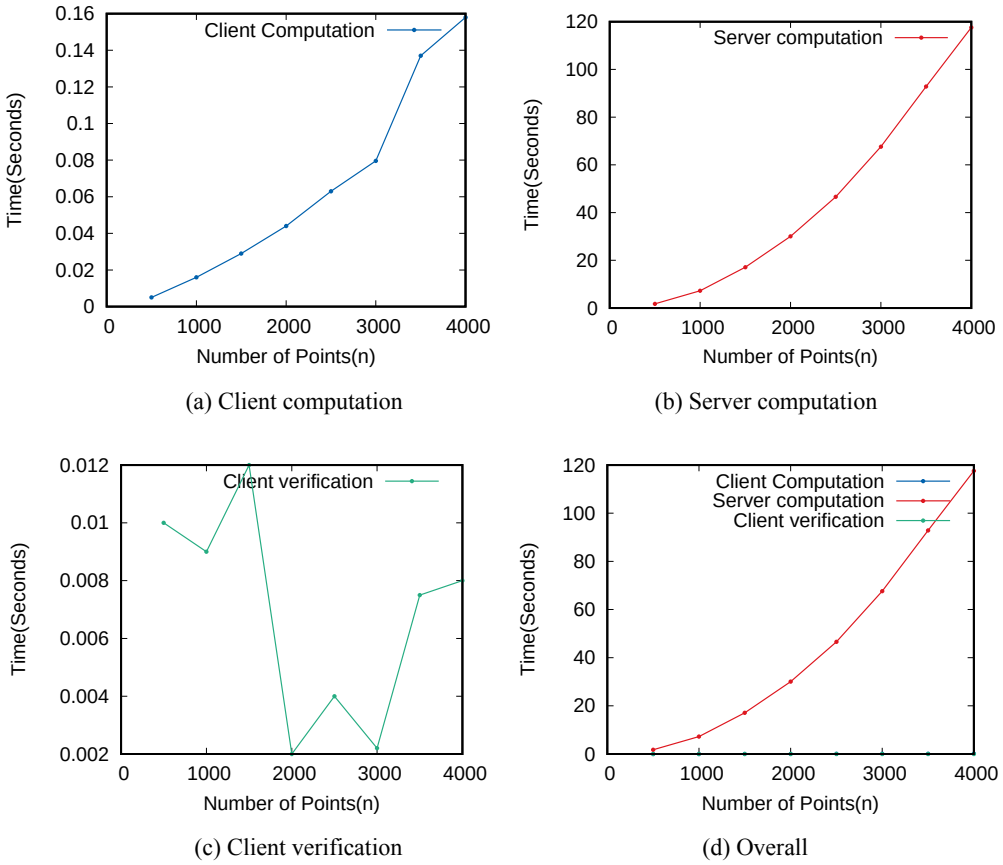
6 Performance analysis and comparison

In this section, we analyse our proposed scheme's performance and compare the proposed scheme with the existing scheme. We have implemented algorithm for client computation, server computation and client verification in Python language. All experiments were run on Windows 10 operating system with an AMD A8-7410@2.5 GHz CPU, 8 GB RAM. Experiments were performed for the varying values of n and r. In particular, the experiments were conducted for $n = 500,1000,$ $1500,2000,2500,3000,3500,4000$ and $r = \frac{n}{10}, \frac{n}{8}, \frac{n}{6}, \frac{n}{4}$, where n is actual number of points present in the input set and r is the extra points added to the input set.

We have performed experiments for each value of r and for every r, the n ranges from 500 to 4000. Graph for $r = \frac{n}{10}, \frac{n}{8}, \frac{n}{6}, \frac{n}{4}$ is shown in Figure 1, Figure 2, Figure 3 and Figure 4 respectively. In each of the figure $x-axis$ shows the number of input data points and $y-axis$ shows the computational time in seconds. For $r = \frac{n}{10}$, Figure 1(a) shows the client computation. It is the time taken by the client to mask the input points and outsource the points to the CSP. Figure 1(b) shows the CSP computation time. This is the time taken by the server to compute Euclidean distance matrix and sort the Euclidean distances. Figure 1(c) shows the client verification time. It is the time taken by the client to verify whether result returned by the server is correct or not. Figure 1(d) shows together the client computation, server computation and the client verification time. Since client computation and client verification time are very close they are overlapping in Figure 1(d).

Figure 2(a), Figure 2(b), Figure 2(c) and Figure 2(d) shows the cost of the computations for $r = \frac{n}{8}$.

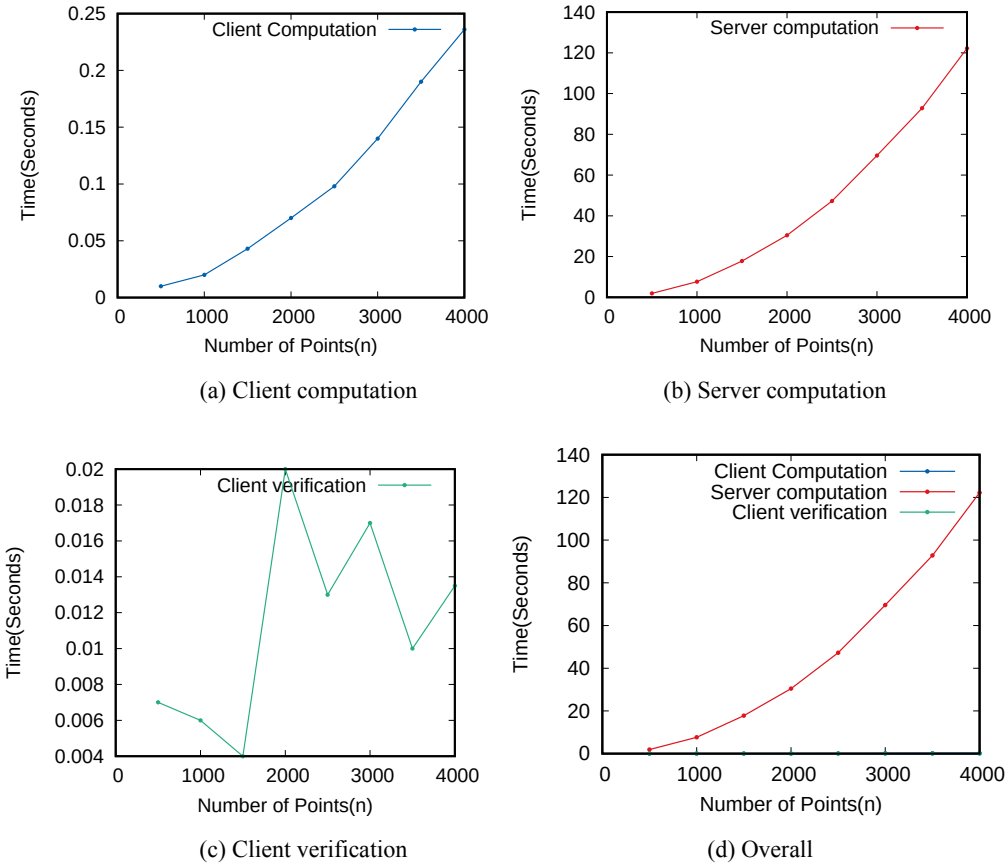
Figure 1 Computation cost for $r = n/10$



(a) Client computation (b) Server computation

(c) Client verification (d) Overall

Figure 2 Computation cost for $r = n/8$



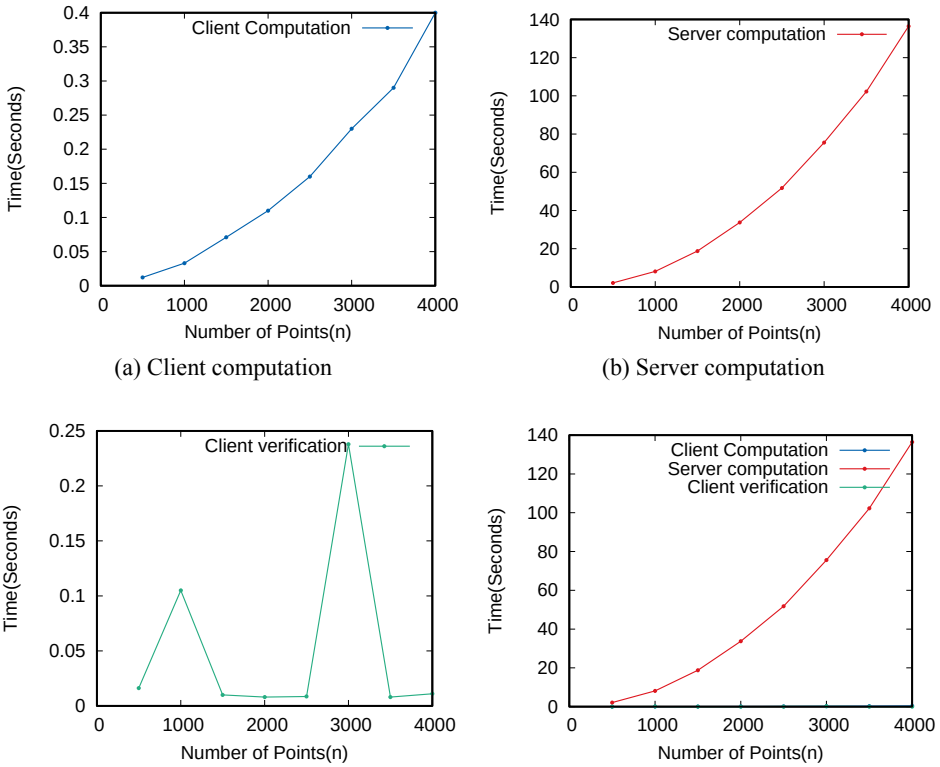
(a) Client computation (b) Server computation

(c) Client verification (d) Overall

Figure 3(a), Figure 3(b), Figure 3(c) and Figure 3(d) shows the cost of the computations for $r = \frac{n}{6}$.

Figure 4(a), Figure 4(b), Figure 4(c) and Figure 4(d) shows the cost of the computations for $r = \frac{n}{4}$.

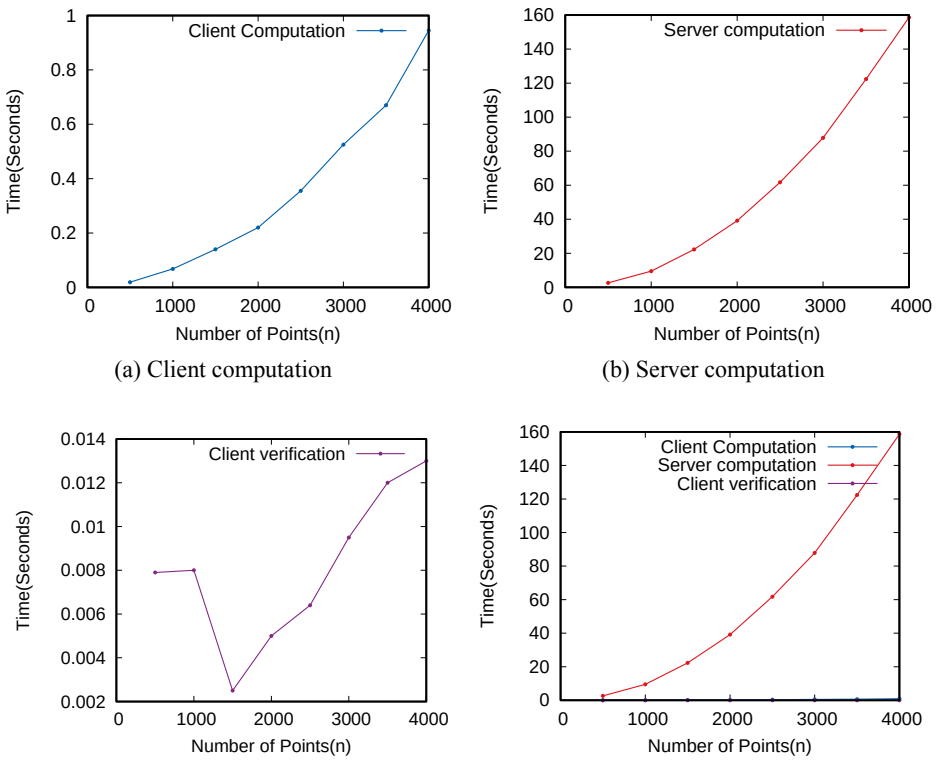
Figure 3 Computation cost for $r = n/6$



(a) Client computation (b) Server computation

(c) Client verification (d) Overall

Figure 4 Computation cost for $r = n/4$



(a) Client computation (b) Server computation

(c) Client verification (d) Overall

It can be observed that the client verification time is very less compared to the original amount of computation done by the CSP. Four experiments are carried out for different values of r since, with the increase in the value of r, the probability of server cheating decreases. So, there is a trade-off between the computation at the CSP and the server cheating probability. However, for practical applications, it can be seen that, for the small change in the value of r, there is a significant decrease in the probability with only less amount of extra computation for verification. When $r = \frac{n}{10}$ client computation time ranges between 0 to 0.16 seconds, server computation time ranges between 0 to 120 seconds, client verification time ranges between 0 to 0.012. When $r = \frac{n}{4}$ client computation time ranges between 0 to 1 seconds, server computation time ranges between 0 to 160 seconds, client verification time ranges between 0 to 0.013 seconds. Increasing the value of r does not affect client computation and client verification time too much, but server computation time is increased. Hence the probability of server cheating can be reduced to negligible by increasing r value with a negligible increase in client computation and client verification.

6.1 Server cheating probability

We compare the proposed scheme's cheating probability with the existing scheme by Kuruba et al. (2016). Server cheating probability comparison is shown in Figure 5. In Figure 5, $x-axis$ shows the extra number of points (r) and $y-axis$ shows server cheating probability. Graph is plotted for $n=100$ and $r=1,2,3,4,5,6$. It can be seen that for the proposed scheme, server cheating probability decreases as the number of extra points (r) added increases but in the existing scheme by Kuruba et al. (2016) cheating probability remains constant. For practical applications, with very small r, our proposed scheme almost has a negligible probability.

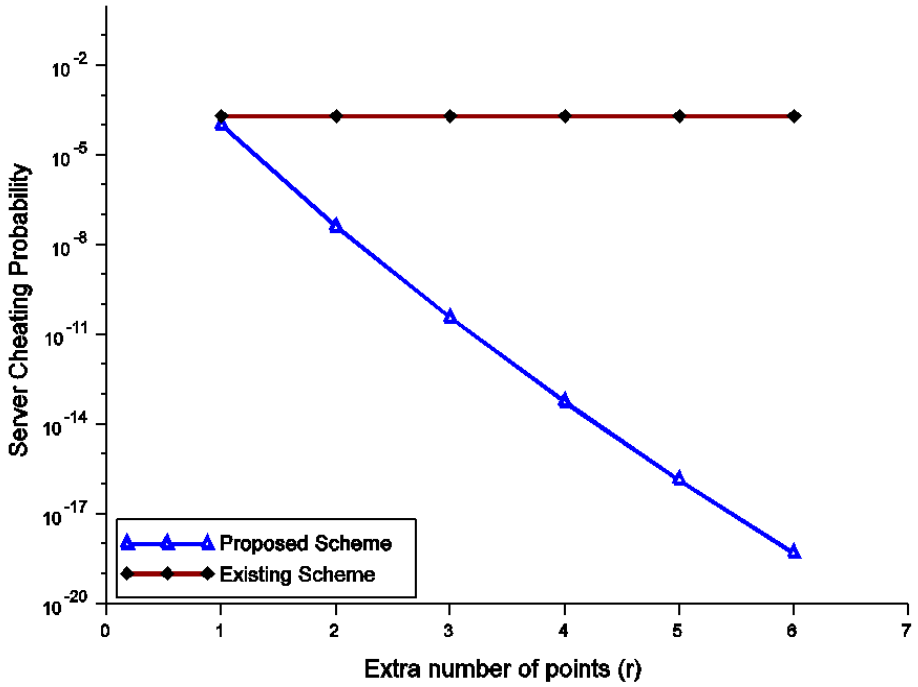
6.2 Comparison of proposed scheme and existing scheme

In this section, we compare the proposed scheme with the existing scheme by Kuruba et al. (2016) with respect to the computation cost and the server cheating probability. Comparison of Kuruba et al. (2016) scheme and proposed scheme is shown in Table 3. The scheme by Kuruba et al. (2016) uses homomorphic encryption whereas the proposed scheme does not make use of homomorphic encryption. Also, Kuruba et al. (2016) scheme has assumed a two-server model, and the proposed scheme uses a single server model where the server is not trusted. The server cheating probability is more in Kuruba et al. (2016) scheme, and in the proposed scheme, the server cheating probability depends on r, and as r increases, the probability decreases.

Table 3 Comparison of proposed scheme and existing scheme

Schemes	*Kuruba et al. (2016) scheme*	*Proposed scheme*
Homomorphic Encryption	Yes	No
Server Model	Two-server model	Single server model
Probability of Server Cheating	$\frac{2}{n(n-1)}$	$\frac{r!r!(n-r)!}{(n+r)!}$

Figure 5 Server cheating probability



In Table 4, we have shown the proposed scheme's improvements over the existing scheme. The scheme by Kuruba et al. (2016) uses a two-server model; hence it is not collusion safe. Whereas the proposed scheme uses a single server model; hence it is collusion safe. In Kuruba et al. (2016) scheme, each point $p_i = (x_i, y_i)$ is split into two parts $p_{i_1} = (x_{i_1}, y_{i_1})$ and $p_{i_2} = (x_{i_2}, y_{i_2})$ and p_{i_1}, p_{i_2} are sent to first cloud server and second cloud server respectively. So for n input points, the minimum number of points that should be sent to CSP is $(n+n=2n)$, whereas in the proposed scheme the minimum number of points that should be sent to CSP is $(n+r)$, where $r \geq 1$. In Kuruba et al. (2016) scheme, server cheating probability remains constant, whereas in proposed scheme server cheating probability can be decreased by increasing value of r.

Table 4 Proposed scheme improvements over the existing scheme

Schemes	*Kuruba et al. (2016) scheme*	*Proposed scheme*
Collusion safe	No	Yes
Minimum number of points to be sent to CSP	$n+n$	$n+r$
Server Cheating Probability	Remains constant	Decreases by increasing r

7 Conclusion

Privacy-preserving verification schemes have been proposed to verify result of outsourced Euclidean distance and the closest pair of points returned by the single untrusted cloud service provider. The proposed scheme adds the flexibility of achieving the tradeoff between the client verification time and the server cheating probability. The proposed scheme allows the applications to achieve the negligible server cheating probability being collusion safe. Compared to the existing scheme which is not collusion safe and with constant server cheating probability, the proposed scheme is the choice for diverse applications employing closest pair of point computations because of its simplicity, flexibility and tendency to reduce the server cheating probability to negligible. Also, the server computation time is constant and more in the existing scheme compared to the proposed scheme.

References

Atallah, M.J. and Frikken, K.B. (2010) 'Securely outsourcing linear algebra computations', *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pp.48–59.

Benjamin, D. and Atallah, M.J. (2008) 'Private and cheating-free outsourcing of algebraic computations', *Sixth Annual Conference on Privacy, Security and Trust*, pp.240–245.

Erfan, F. and Mala, H. (2020) 'Secure and efficient publicly verifiable outsourcing of matrix multiplication in online mode', *Cluster Computing*, Vol. 23, pp.2835–2845.

Kuruba, C., Gilbert, K., Sidhaye, P., Pareek, G. and Purushothama, B.R. (2016) 'Outsource-secured calculation of closest pair of points', *Security in Computing and Communications - 4th International Symposium, Proceedings*, pp.377–389.

Le, W. and Li, F. (2012) 'Query access assurance in outsourced databases', *IEEE Transactions on Services Computing*, Vol. 5, No. 2, pp.178–191.

Lei, X., Liao, X., Huang, T., Li, H. and Hu, C. (2013) 'Outsourcing large matrix inversion computation to a public cloud', *IEEE Transactions on Cloud Computing*, Vol. 1, pp.78–87.

Li, P., Xu, H. and Guo, S. (2013) 'Public verification of outsourced computation of polynomial functions', *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp.776–780.

Li, Z., Li, W., Jin, Z., Zhang, H. and Wen, Q. (2019) 'An efficient ABE scheme with verifiable outsourced encryption and decryption', *IEEE Access*, Vol. 7, pp.29023–29037.

Murugesan, M., Jiang, W., Chris, C., Luo, S. and Vaidya, J. (2010) 'Efficient privacy-preserving similar document detection', *The VLDB Journal*, Vol. 19, No. 4, pp.457–475.

Sheng, G., Wen, T., Guo, Q. and Yin, Y. (2013) 'Verifying correctness of inner product of vectors in cloud computing', *Proceedings of the 2013 International Workshop on Security in Cloud Computing*, pp.61–68.

Shuai, L., Longxia, H., Anmin, F. and Yearwood, J. (2017) 'CExp: secure and verifiable outsourcing of composite modular exponentiation with single untrusted server', *Digital Communications and Networks*, Vol. 3, No. 4, pp.236–241.

Sion, R. (2005) 'Query execution assurance for outsourced databases', *Proceedings of the 31st International Conference on Very Large Data Bases*, pp.601–612.

Xixun, Y., Zheng, Y. and Rui, Z. (2019) 'Verifiable outsourced computation over encrypted data', *Information Sciences*, Vol. 5, No. 2, pp.178–191.

Xue, T., Ren, Y. and Feng, G. (2018) 'An IBE scheme with verifiable outsourced key generation based on a single server', *IETE Technical Review*, Vol. 35, pp.97–105.

Ye, J., Xu, Z. and Ding, Y. (2016) 'Secure outsourcing of modular exponentiations in cloud and cluster computing', *Cluster Computing*, Vol. 19, pp.811–820.

Zhang, J., Yang, Y. and Wang, Z. (2016a) 'Outsourcing large-scale systems of linear matrix equations in cloud computing', *IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pp.438–447.

Zhang, S., Li, H., Jia, K., Dai, Y. and Zhao, L. (2016b) 'Efficient secure outsourcing computation of matrix multiplication in cloud computing', *2016 IEEE Global Communications Conference (GLOBECOM)*, pp.1–6.

Zhu, Y., Fu, A., Yu, S., Yu, Y., Li, S. and Chen, Z. (2018) 'New algorithm for secure outsourcing of modular exponentiation with optimal checkability based on single untrusted server', *IEEE International Conference on Communications (ICC)*, pp.1–6.