Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 28 (2025), No. 5-A, pp. 1825–1834
DOI : 10.47974/JDMSC-2182

Privacy preserving protocols for encrypted data comparison in semi-honest cloud environment

Vijayendra Sanjay Gaikwad [†]
P. G. Department of Computer Science & Engineering
Sant Gadge Baba Amravati University
Amravati 444602
Maharashtra
India

and

SCTR'S Pune Institute of Computer Technology
Pune 411043
Maharashtra
India

Kishor H. Walse *
Sant Bhagwanbaba Kala
Mahavidyalaya Sindkhed Raja
Buldana 443203
Maharashtra
India

Mohammad Atique [§]
P. G. Department of Computer Science & Engineering
Sant Gadge Baba Amravati University
Amravati 444602
Maharashtra
India

[†] *E-mail:* vij711@gmail.com
* *E-mail:* walse.kishor@gmail.com (Corresponding Author)
[§] *E-mail:* mohammadatique@sgbau.ac.in

Abstract

Currently, with third-party cloud providers handling data mining tasks while securely processing sensitive data has become critical. These providers may exploit intermediate data during computation, posing privacy risks. In privacy-preserving classification, we address a secure comparison problem where inputs a and b are encrypted, ensuring no plaintext is exposed to the cloud. Our protocol, designed for a semi-honest model, allows only the end user to learn whether $a > b$, keeping both inputs hidden from cloud parties. While initially targeting single comparisons, our protocol scales to multiple encrypted inputs to identify the maximum value securely. It thus serves as a foundational tool for secure data processing. Additionally, we introduce a second protocol utilizing homomorphic encryption's additive property to compare encrypted values efficiently, reducing both computational and communication overhead. Both protocols avoid circuit-based methods and operate in a fixed number of rounds, enhancing performance.

Subject Classification: *Primary 93A30, Secondary 49K15.*

Keywords: *Privacy preserving classification, Encrypted data comparison, Homomorphic encryption.*

1. Introduction

Encrypted value comparison is a fundamental component in many privacy-preserving classification applications, especially in domains like healthcare, finance, and social networks. Efficient protocols are essential for enhancing both security and performance. The classic Yao's Millionaire Problem (YMP) [1], introduced nearly four decades ago, addresses secure comparison between two parties without revealing additional information. However, YMP is limited to single comparisons.

Consider a scenario with a database owner (DO) who outsources a database with m attributes and n records to the cloud, and an authorized user (A) who submits encrypted queries to perform k-Nearest Neighbors (kNN) classification [2]. As both the data and queries are sensitive, neither DO nor A wishes to expose information to the cloud or each other. A core step in kNN is computing the squared Euclidean distance between the query and each record, which can be done securely using partial homomorphic encryption [3].

After computing encrypted distances, identifying the smallest one is essential to determine nearest neighbors. However, since the cloud cannot decrypt the values, a secure protocol is needed to find the record with the minimum encrypted distance without revealing any actual values. We refer to this challenge as the "encrypted data comparison" problem and propose three solutions tailored to its variants.

Our protocols serve as key components for secure data processing [4], designed specifically for the "greater-than" operation but adaptable to "equals" and "less-than" with minor modifications. Even modest efficiency improvements in this operation can significantly boost overall performance when applied to large encrypted datasets.

The rest of this paper is organized as follows: Section 2 reviews related work, Section 3 details our proposed solutions, Section 4 presents the empirical results and comparative analysis, and Section 5 concludes with future research directions.

2. Literature Review

The work by Jiang et al. in [5] utilizes an Order-Revealing Encryption (ORE) scheme that allows secure range queries on encrypted data. The method employs a specialized encryption function that enables the comparison of encrypted values without revealing the actual data. The ORE ensures that the encrypted data maintains a specific order, facilitating range queries while preserving privacy. However, there is an issue with the leakage of order information. The recent work by Chen et al. in [6] introduces a Secure Multi-Party Computation (SMPC) framework based on a Partial Homomorphic Encryption (PHE) scheme [3]. Their protocol uses incomplete re-encryption to allow the cloud servers to perform secure data comparisons. The approach in has lower processing and communication costs than existing solutions. Additionally, Zero-Knowledge Proof (ZKP) is implemented to prevent malicious behavior by clients to manipulate the protocol. The re-encryptions of differences however, adds to the protocol's computational cost and hence, is an overhead.

Tao et al. in [7] have presented an even improved version of the Damgård-Geisler-KrØigaard (DGK) comparison protocol [8], designed to enhance security and efficiency in secure multi-party computation (SMPC) environments. The protocol has a lower computational overhead compared to fully homomorphic encryption schemes [9]. Major problem with SMPC protocols is complex cryptographic operations that require significant computational resources and frequent communication between parties resulting in high communication overhead, making them less practical for large datasets.

Yao [1] introduced secure two-party computations and later extended to multi-party computation. According to [10], while binary logic

techniques seem adaptable, cost of these approaches is dictated by logic design's magnitude.

In [11] and [12], a variation in YMP, which uses two input vectors, one each for binary forms of a and b. However, we focus on comparison of integer values that are in ciphertext forms. Additionally, proposed approach does not rely on circuits, making it simpler and faster to implement. Many other recent protocols are built upon YMP's basic solution, focusing on securely determining whether a>b or a<b where a is the first party's private input and b is the second party's private input, however in our case both private inputs belong to single party and the comparison operation is to be performed in outsourced cloud environment.

3. Proposed Methodology for Encrypted Data Comparison

Our problem setting involves operating within a genuine-but-curious scenario, where the two cloud platforms involved adhere strictly to the protocol specifications. However, these platforms might try inferring some knowledge through intermediate data analysis as they receive data during the protocol's execution. With our protocols utilizing additive homomorphic property, we ensures that while the protocols are followed correctly, data privacy is maintained by preventing any cloud from gaining extra information beyond what the protocol allows. To illustrate the problem setting, assume any user wants to compare two values a and b privately. User possesses its own public key and suppose Epk-U () is used for performing encryption using the public key of user and Dsk-U () is used for performing decryption using user's secret key. User encrypts a and b with its public key using partial homomorphic encryption scheme to get Epk-U (a) and Epk-U (b). User then outsources these encrypted values to first cloud which collaborates with the second cloud to perform the encrypted data comparison task. User also outsources the public key to both clouds and the secret key to only second cloud. Our goal is to develop a protocol that determine whether a<b or a>b and conveys it directly to the user, keeping original a and b undisclosed to both clouds.

We propose two approaches in this paper as solutions to the "encrypted data comparison" problem. Our protocols are based on the use of additive properties from the partial homomorphic encryption scheme in [3].

Algorithm-A: "Protocol for comparing two outsourced encrypted values"

1. User outsources two encrypted values $E_{pk\text{-}U}(a)$, $E_{pk\text{-}U}(b)$ to Cloud-1.
2. Cloud-1 randomly chooses a number i (where $i \in (1,\ldots, 100)$) to perform the following computations:

$$R' = E_{pk\text{-}U}(i)$$

$$A' = E_{pk\text{-}U}(a - i) = E_{pk\text{-}U}(a) * R'^{N-1} \quad (1)$$

$$B' = E_{pk\text{-}U}(b - i) = E_{pk\text{-}U}(b) * R'^{N-1} \quad (2)$$

3. Cloud-1 sends A' and B' to Cloud-2.
4. Cloud-2 then decrypts A' and B' with $D_{sk\text{-}U}$ and privately computes the values of X_t as:

$$X_t = D_{sk\text{-}U}(A') + t \text{ , where } t \text{ is from 1 to 100}$$

$$\text{Hence, at } t = i,\ X_i = D_{sk\text{-}U}(A') + t = a$$

5. Cloud-2 also privately computes the values of Y_t as:

$$Y_t = D_{sk\text{-}U}(B') + t \text{ , where } t \text{ is from 1 to 100}$$

$$\text{Hence, at } t = i,\ Y_i = D_{sk\text{-}U}(B') + t = b$$

6. Any random number r is generated by Cloud-2 to compute the following,

$$Z_t = X_t + r \text{ , for all } t$$

$$Z'_t = Y_t + r \text{ , for all } t$$

7. Cloud-2 sends the randomized lists Z_u and Z'_u to Cloud-1 and r to the user.
8. Cloud-1 receives the randomized lists Z_u and Z'_u, then compares the i^{th} elements (Z_i and Z'_i) of both randomized lists without realizing the original values of a and b and sends the larger randomized value, L_r to the user.
9. User receives r from Cloud-2, L_r from Cloud-1 and computes the original larger value as,

$$L = L_r - r \quad (3)$$

The next protocol is an extension of the first protocol which effectively solves the "encrypted data comparison" problem with multiple encrypted values to be compared in the outsourced cloud environment.

Algorithm-B: "Protocol for comparing multiple outsourced encrypted values"

1. User outsources n encrypted values $E_{pk\text{-}U}(a_1)$, …, $E_{pk\text{-}U}(a_n)$ to Cloud-1.
2. Cloud-1 randomly chooses a number i (where $i \in (1,\ldots, 100)$) to perform the following computations:

$$R' = E_{pk\text{-}U}(i)$$

$$A'_1 = E_{pk\text{-}U}(a_1 - i) = E_{pk\text{-}U}(a_1) * R'^{N\text{-}1}$$

$$:$$

$$:$$

$$A'_n = E_{pk\text{-}U}(a_n - i) = E_{pk\text{-}U}(a_n) * R'^{N\text{-}1}$$

3. Cloud-1 sends $A'_1, \ldots, A'_n$ to Cloud-2.
4. Further steps are similar to steps (3) to (6) of previous protocol.

The next protocol directly leverages addition property from Paillier encryption scheme [3], which allows for secure computation on encrypted data without decryption. We operate within the semi-honest model, ensuring that while the protocol is correctly followed, data privacy is maintained by preventing any party (any cloud platforms) from gaining extra information beyond what the protocol allows.

Algorithm-C: "Protocol based on direct homomorphic randomization"

1. User encrypts a and b and outsources encrypted values $E_{pk\text{-}U}(a)$, $E_{pk\text{-}U}(b)$ to Cloud1.
2. Cloud1 chooses randomly generated integer R as blinding factor and encrypts it using user's public key to obtain randomization component, $E_{pk\text{-}U}(R)$
3. Cloud1 then performs masked randomization of a and b using homomorphic operations:

$$E_{pk\text{-}U}(a+R) = E_{pk\text{-}U}(a) * E_{pk\text{-}U}(R) \quad (4)$$

$$E_{pk\text{-}U}(b+R) = E_{pk\text{-}U}(b) * E_{pk\text{-}U}(R) \quad (5)$$

4. Cloud1 sends the randomized encrypted inputs $E_{pk\text{-}U}(a+R)$ and $E_{pk\text{-}U}(b+R)$ to Cloud2 and the blinding factor R is sent to the user.
5. Cloud2 then decrypts $E_{pk\text{-}U}(a+R)$ and $E_{pk\text{-}U}(b+R)$ with $D_{sk\text{-}U}$:

$$a' = D_{sk\text{-}U}\,(\,E_{pk\text{-}U}(a+R))$$

$$b' = D_{sk\text{-}U}\,(\,E_{pk\text{-}U}(b+R))$$

6. Cloud2 compares the decrypted-but-randomized values a' and b':
   if $a' > b'$, then
   $r' = a'$
   send r' to the user.
   else
   $r' = b'$
   send r' to the user.
7. User receives the randomized resultant value r' from Cloud2 and the blinding factor R from Cloud1and eliminates the randomization component as:

$$r = r' - R \tag{6}$$

This approach enhances the efficiency of the protocol and also maintains practicality for privacy-preserving data comparisons. An extended version of this protocol can effectively solves the "encrypted data comparison" problem with multiple encrypted values to be compared in the outsourced cloud environment.

4. Empirical Results and Comparative Analysis

We carried out multiple experiments and measured the performance of our protocols on an Intel® CORE i5® eight-core CPU with a 2.5 GHz clock speed and 8 GB of RAM, operating on Ubuntu 22.04 LTS. We implemented our protocols using the Paillier cryptosystem [1] via the Python 'phe' library, which supports partially homomorphic encryption. Given the computational overhead of homomorphic operations compared to plaintext processing, we evaluated the encryption and decryption times for integer values using 2048-bit public and private keys.

The encryption time for a 5-digit integer value is around 0.45 seconds and decryption time is around 0.15 seconds. So, considering our problem setting where a user wants to compare two integer values (each of 5-digits) privately, the execution time required by our Algorithm-A is 0.89 seconds which is reasonable. Algorithm-B which is based on first approach and is extension of Algorithm-A for comparing multiple encrypted values, was run with total encrypted values for comparison i.e. k from 10 to 100 to measure its performance. Table 1 shows the execution time required by

Table 1

Execution time in seconds required for Algorithm-B by varying k and N

	N = 100	**N= 300**	**N = 700**	**N =1000**	**N=5000**
k =10	4.7	4.9	5.2	5	5.05
k = 25	12.54	12.52	12.68	12.64	12.68
k = 50	24.8	25.2	25.05	24.64	24.76
k = 75	38.18	37.26	36.74	36.63	36.91
k = 100	50.12	50.2	49.71	49.12	49.6

our Algorithm-B for varying values of k from 10 to 100 and N (i.e. range size) from 100 to 5000. Notably, it has been observed that the protocol execution time stabilizes once N exceeds 700 for any given value of k.

Table 1 shows that when number of encrypted values to be compared (i.e. k) is increased the execution time also increases. It also depicts the plateau in execution time for all values of k when N is any value chosen beyond 700. Hence, choosing a large value for N so as to mitigate its prediction risk and provide better privacy, will not have an impact on the protocol's performance.

Algorithm-C uses our second approach and the execution time required by it is 0.5 seconds which is significantly less (almost half) than that of Algorithm-A. We also experimented with Algorithm-C to compare multiple encrypted values and measured the performance. Table 2 shows the execution time required for comparing multiple encrypted values using second approach by varying values of k from 10 to 100. It depicts the linear increase in the execution time with increasing values of k. We clearly observe that the execution time for Algorithm-C is significantly less when compared with that of Algorithm-B for all values k. Hence, Algorithm-C is communication-wise and computationally more efficient.

Table 2

Execution time required for comparing multiple encrypted values using second approach with varying k

k **=10**	k **= 25**	k **= 50**	k **= 75**	k **= 100**
1.3	2.9	5.74	8.33	10.96

5. Conclusion and Future Work

Our approaches are based on the effective use of additive properties from the partial homomorphic encryption scheme. As per mentioned

computation complexities, both approaches require linear execution time. However, since the overall computational complexity of second method is dependent on total encrypted values to be compared and therefore it is communication-wise and computationally more efficient.

Both of our approaches operate in a fixed number of rounds and avoid using conventional circuit processing methods, hence have improved efficiency. Our proposed protocols are simple to understand as they do not demand a deep knowledge of cryptography or number theory, easy to implement and also efficient, requiring minimal time. Hence, being straightforward practical solutions, proposed protocols can serve as crucial building blocks, offering secure comparison capabilities in scenarios of secure data processing.

References

[1] A. C. Yao, "Protocols for secure computations," 23rd Annual Symposium on Foundations of Computer Science. Chicago, USA, pp. 160-164 (1982). doi: 10.1109/SFCS.1982.38

[2] T. Cover and P. Hart, "Nearest neighbor pattern classification," in *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21-27 (1967), doi: 10.1109/TIT.1967.1053964.

[3] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes," Advances in Cryptography- EUROCRYPT '99. Prague, Czech Republic, pp. 223-238 (1999). doi: https://doi.org/10.1007/3-540-48910-X_16.

[4] Vishal Kapoor, Pascal Poncelet, François Trousset, Maguelonne Teisseire, et al., "Privacy-preserving sequential pattern mining in distributed databases," Proceedings of the 15th ACM international conference on Information and knowledge management. Arlington, Virginia, USA (2006).

[5] Leqi Jiang, Yi Cao, Chengsheng Yuan, Xingming Sun, Xiaoli Zhu, "An effective comparison protocol over encrypted data in cloud computing," in *Journal of Information Security and Applications*, vol. 48, (2019).

[6] Yuling Chen, Junhong Tao, Tao Li, Jiangyuan Cai and Xiaojun Ren, "An Effective Security Comparison Protocol in Cloud Computing," in *CMC-Computers, Materials & Continua*, vol. 74, no. 3, pp. 5897-5914, (2023).

[7] J. Tao, Y. Wu and Y. Chen, "A Secure Comparison Protocol in the Malicious Model," 2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), Espoo, Finland, pp. 332-337 (2022). doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics55523.2022.00079.

[8] T. Veugen, "Improving the DGK comparison protocol," 2012 IEEE International Workshop on Information Forensics and Security (WIFS), Costa Adeje, Spain, pp. 49-54 (2012). doi: 10.1109/WIFS.2012.6412624

[9] C. Gentry, "Fully homomorphic encryption using ideal lattices," in ACM STOC, pp. 169–178 (2009).

[10] Y. Lindell and B. Pinkas, "Privacy-Preserving Data Mining," Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology. (2000).

[11] Juan A. Garay, Berry Schoenmakers, and José Villegas, "Practical and secure solutions for integer comparison," Proceedings of the 10th international conference on Practice and Theory in Public-key Cryptography, Beijing, China. (2007).

[12] B. Schoenmakers Tuyls P "Practical two-party computation based on the conditional gate, " Advances in Cryptology— ASIACRYPT '04, Lecture Notes in Computer Science, Berlin, vol. 3329, pp 119–136 (2004).

Received November, 2024