BLIND: A privacy preserving truth discovery system for mobile crowdsensing

Article

Accepted version

V. Agate, P. Ferraro, G. Lo Re, S.K. Das

Elsevier Journal of Network and Computer Applications
DOI: 10.1016/j.jnca.2023.103811

Publisher: Elsevier

BLIND: A Privacy Preserving Truth Discovery System for Mobile Crowdsensing

Vincenzo Agate[a,b], Pierluca Ferraro[a,b], Giuseppe Lo Re[a,b], Sajal K. Das[c]

[a] *Department of Engineering, University of Palermo*
[b] *Cybersecurity National Lab CINI - Consorzio Interuniversitario Nazionale per l'Informatica*
[c] *Department of Computer Science, Missouri University of Science and Technology*

Abstract

Nowadays, an increasing number of applications exploit users who act as intelligent sensors and can quickly provide high-level information. These users generate valuable data that, if mishandled, could potentially reveal sensitive information. Protecting user privacy is thus of paramount importance for crowdsensing systems. In this paper, we propose BLIND, an innovative open-source truth discovery system designed to improve the quality of information (QoI) through the use of privacy-preserving computation techniques in mobile crowdsensing scenarios. The uniqueness of BLIND lies in its ability to preserve user privacy by ensuring that none of the parties involved are able to identify the source of the information provided. The system uses homomorphic encryption to implement a novel privacy-preserving version of the well-known K-Means clustering algorithm, which directly groups encrypted user data. Outliers are then removed privately without revealing any useful information to the parties involved. We extensively evaluate the proposed system for both server-side and client-side scalability, as well as truth discovery accuracy, using a real-world dataset and a synthetic one, to test the system under challenging conditions. Comparisons with four state-of-the-art approaches show that BLIND optimizes QoI by effectively mitigating the impact of four different security attacks, with higher accuracy and lower communication overhead than its competitors. With the optimizations proposed in this paper, BLIND is up to three times faster than the baseline system, and the obtained Root Mean Squared Error (RMSE) values are up to 42% lower than other state-of-the-art approaches.

Keywords:
Truth discovery, Privacy-Preserving Computation, Mobile Crowdsensing, QoI

1. Introduction

Smartphones are among the most widely used devices worldwide, with more than 7.7 billion smartphone subscriptions by 2027. Their pervasive use has led academic and industrial research efforts over the past fifteen years to focus on the potential offered by such devices, which equip a plethora of embedded sensors, such as cameras, microphones, GPS, gyroscopes, accelerometers, and proximity sensors (Khan et al., 2013).

These sensory technologies, combined with the advanced sensing capabilities of humans, have been used to actively monitor some real world phenomena, giving rise to the Mobile Crowdsensing (MCS) paradigm (Liu et al., 2019). Along with the promising technologies of 6G, which pave the way for a fully connected world by providing ubiquitous wireless connectivity for all (Akyildiz et al., 2020), MCS aims to be a major contributor to applications such as health monitoring (Santani et al., 2018), environmental pollution monitoring (Becnel et al., 2019; Al-Janabi et al., 2020b, 2021), and social networking (Kim et al., 2022).

Since users may often act selfishly or even maliciously (Agate et al., 2021b), and smartphone sensors may be faulty, uncalibrated, or tampered with (Barnwal et al., 2019), the rich information collected by MCS systems may be unreliable. Assessing and improving the quality of information (QoI) available is one of the most important and widely studied challenges of MCS (Zhao et al., 2021). Given that the information collected cannot be assumed to be accurate, Truth Discovery (TD) systems brought a significant contribution in this regard by providing means to obtain high-quality information (Sun et al., 2018). The end goal is to collect and analyze user-generated data from mobile devices. However, this data can be extremely sensitive as it can include personal information such as location, physical activity, browsing habits, etc. Therefore, it is crucial to protect users' privacy during the data collection and analysis process. Indeed, while the collected data, when aggregated, may reveal useful information to share with the community (Alkaim and Al_Janabi, 2020; Al-Janabi and Alkaim, 2020; Al-Janabi and Al-Janabi, 2023), it also poses serious privacy issues that the user is often unaware of (Yu et al., 2022), and that TD systems designers have often ignored. Since participants usually send their observations of phenomena to a cloud server, it is easy to see

Email addresses: vincenzo.agate@unipa.it (Vincenzo Agate), pierluca.ferraro@unipa.it (Pierluca Ferraro), giuseppe.lore@unipa.it (Giuseppe Lo Re), sdas@mst.edu (Sajal K. Das)

how this data could be misused to violate users' privacy without their knowledge.

One of the most promising solutions for ensuring privacy is the use of cryptographic techniques that allow data to be securely processed directly on the user's device, without having to be transferred to a centralized server in plain text. Recently, many researchers have begun to address the problem by proposing privacy-preserving truth discovery systems that are able to estimate aggregate information without revealing the origin of the data, by exploiting homomorphic cryptographic systems and treating data in encrypted form.

However, having user devices perform these costly cryptographic operations presents several challenges in terms of scalability and applicability to real-world scenarios. First, performing many cryptographic operations requires a large amount of energy, which can significantly reduce the battery life of mobile devices. This can be a major concern for users, who may be unwilling to participate in the MCS process if there is a risk of draining their device's battery quickly. In addition, as the number of users increases, the amount of data to be processed also increases, which can make the system unfeasible in practice (Al-Janabi and Alkaim, 2022; Kadhuim and Al-Janabi, 2023; Mohammed and Al-Janabi, 2022).

Despite these challenges, the need to ensure user privacy in an MCS context makes the use of these on-device cryptographic techniques essential. The goal is therefore to strike a balance between privacy protection and computational and energy efficiency. This requires the development of new cryptographic techniques that are both secure and efficient, or the use of alternative approaches that minimize the need for costly cryptographic operations performed directly on users' mobile devices. Our solution does both, improving on existing systems by optimizing cryptographic operations and involving users as little as possible in cryptographic computations, in order to make the process realistic and scalable.

To this end, we propose BLIND, a novel privacy-preserving truth discovery system for mobile crowdsensing, offering a new advanced model for optimizing the QoI by clustering user observations. This happens in a completely privacy-preserving way, while keeping secret both the users' reports, their association with respective clusters, and the identity of outliers. Aggregated information from multiple clusters may be significant in itself or may be further combined to obtain a single truth, depending on the application scenario.

A key limitation of many existing works is that they require all users to participate in the truth discovery process at the same time and to remain connected throughout the process, as they must constantly interact with each other and the server. In a mobile crowdsensing context, the idea that all users must remain connected during the entire process is unrealistic, and this can be a serious problem. Mobile users have unpredictable behavior, and their device may lose connection, be turned off, or run out of battery power. BLIND, on the other hand, minimizes the operations that users have to perform on their mobile devices, which has significant implications for a privacy-preserving system in a mobile scenario. By limiting user involvement to the initial encryption of data, the system reduces the computational load on mobile devices. This is much more sustainable in terms of energy and computational cost, and preserves the user's battery and device resources.

BLIND is also very scalable and can handle a large number of users without the risk of overloading their devices; the extra cryptographic operations only impact the server, which can be scaled both vertically and horizontally as needed.

Another key advantage of BLIND is being open source. To the best of our knowledge, we are the first to release an open source version of a privacy-preserving truth discovery system. This is particularly relevant for privacy-preserving TD applications, since gaining user trust is critical to the success of such systems, as discussed in (Zigomitros et al., 2020). Moreover, being open source, BLIND is easily extensible and can be tailored to different application scenarios. Our source code is freely accessible on GitHub[1].

The proposed solution has been extensively tested on a real-world dataset and the results obtained clearly demonstrate how our system outperforms its direct competitors in terms of achievable accuracy. Finally, scalability tests performed on synthetic data demonstrate its suitability in applications with a large number of users.

The main contributions are summarized as follows.

- This work introduces BLIND, a novel truth discovery system that improves the QoI by clustering observations and discarding unreliable data while preserving users' privacy.
- A novel secure privacy-preserving protocol is provided to cluster user data in a Mobile Crowdsensing system, without revealing confidential information.
- It is demonstrated that BLIND is secure and that no information about users is disclosed to any of the parties involved in the protocol.
- Extensive experiments based on real data are conducted to evaluate the performance of our system in terms of accuracy and to compare it with other state-of-the-art approaches in realistic scenarios that include four different security attacks. Experimental results show that BLIND outperforms its competitors.
- A further evaluation on the communication overhead and scalability of BLIND (both server-side and client-side) is proposed using a synthetic dataset to test the system under challenging conditions.

[1] https://github.com/ndslab-unipa/BLIND

Table 1: Comparison of the main privacy preserving truth discovery systems proposed in the literature with BLIND.

	Homomorphic cryptography	Offline computation	Low client-side effort	Open source
PPTD (Miao et al., 2019)	✓	-	-	-
L-PPTD (Miao et al., 2017)	✓	✓	-	-
EPTD (Xu et al., 2017)	✓	✓	-	-
LPTD-I (Zhang et al., 2019b)	✓	-	-	-
RPTD-I (Zhang et al., 2019a)	✓	-	✓	-
PPTDS-I (Zhang et al., 2019c)	-	✓	✓	-
EPTD-I (Zhang et al., 2020a)	-	-	✓	-
SATE (Zhang et al., 2020b)	✓	✓	✓	-
BLIND (proposed system)	✓	✓	✓	✓

The remainder of this paper is organized as follows. Section 2 summarizes most relevant work in literature. Section 3 presents our problem formulation and system model and Section 4 provides some background on PPC techniques. Our system is described in detail in Section 5. Section 6 formally demonstrate the security and privacy-preserving characteristics of BLIND. Section 7 presents our experimental results and Section 8 discusses strengths and limitations of the system and provides our insights on the experimental evaluation. Finally, Section 9 concludes the paper.

2. Related Work

Over time, researchers in the field of Mobile Crowdsensing (MCS) have attempted to address issues regarding three main aspects: how to define the concept of QoI, how to encourage the submission of high-quality observations, and how to estimate and improve the quality of the information received (Liu et al., 2019). MCS systems are used for an extremely wide range of applications, such as health monitoring (Santani et al., 2018), traffic monitoring (Bhattacharjee et al., 2018), environmental pollution monitoring (Becnel et al., 2019), and vehicular networks (Cheng et al., 2022). Given the extremely varied nature of the data collected (Al-Janabi and Al-Janabi, 2022; Al-Janabi et al., 2023, 2020a; Al-Janabi and Al-Barmani, 2023), it is difficult to agree on a definition of QoI, and much of the research has focused on this task. Moreover, given that information collected by MCS systems may be affected by selfish or malicious behavior of the participants (Shehada et al., 2018) and that data provided by sensors may be inaccurate or tampered with, it is necessary to optimize the QoI of the observations provided by users.

Another issue that researchers have addressed is the need to design incentive mechanisms that keep users interested in participating to MCS systems (Luo et al., 2017), while also encouraging them to share high-quality accurate information (Restuccia et al., 2018b). Much of the recent research in MCS systems has specifically addressed detecting and incentivizing reliable participants while optimizing QoI (Khan et al., 2019), often overlooking other key aspects such as maintaining user privacy.

One of the first truth discovery frameworks proposed in literature, called CRH (Li et al., 2016), models the issue of conflict resolution in heterogeneous data (both categorical and continuous) as an optimization problem. To solve this problem, the authors proposed an iterative algorithm, which inspired many subsequent truth discovery systems over the past years. Although this system is regarded as pioneristic for QoI optimization and represents the baseline to compete with, in terms of MCS accuracy, it completely overlooks a critical issue, namely the privacy of participants.

2.1. Privacy Preserving Truth Discovery Systems

Many studies show that a small amount of data or features derived from social networks (Liao et al., 2018) or from demographic reports, opportunely filtered, is sufficient to uniquely identify a citizen (Cecaj et al., 2016). Neverova et al. (2016), for example, show that even human kinematics, perceived through sensors equipped on common smartphones, convey important information about the user's identity. In addition, the same data received from users can often be exploited without their knowledge for targeted advertising (Chorppath and Alpcan, 2013). In other contexts, data submitted by users may be extremely sensitive and maintaining their secrecy is paramount, as in the case of e-Voting systems (Agate et al., 2021a). In this regard, a regulatory gap is being closed recently, which has often left companies free to exploit user data indiscriminately and for a variety of purposes out of context. Despite regulatory efforts, we believe that a different approach should be used to collect this important information without putting users' privacy at risk. General approaches to ensure privacy protection in pervasive systems are described by Bettini and Riboni (2015).

While many works have been concerned with discovering the truth in case of conflicting information (Restuccia et al., 2018a), only a few exploits Privacy-Preserving Computation (PPC) techniques to ensure users' privacy (Li et al., 2013). In particular, homomorphic encryption allows a third party to perform operations on encrypted data, without becoming aware of users' confidential information. As a result, users can assign heavy computations to cloud servers without compromising their pri-

Table 2: Comparative analysis of the proposed approach with previous works.

System name	Techniques	Dataset	Evaluation measures	Advantages	Disadvantages
CRH(Li et al., 2016)	Optimization framework, Weighted deviation minimization	Weather forecast, Stock, Flight	Error rate, Mean Normalized Absolute Distance	Estimation of source reliability, Adaptation for large-scale data	Not privacy preserving
PPTD(Miao et al., 2019)	Homomorphic cryptography, threshold Paillier	Synthetic, custom realistic	MAE, RMSE, running time	Parallelization with MapReduce techniques	Computational burden for users, assumes non-colluding users
L-PPTD(Miao et al., 2017)	Homomorphic cryptography, Paillier	Synthetic, homemade realistic	Error rate, RMSE	Low user effort	User data is not encrypted when sent
EPTD(Xu et al., 2017)	Homomorphic cryptography, one-way trapdoor function	Not specified	Running time, communication overhead	CRH accuracy estimation	Involves users in all phases, assumes non-colluding users
LPTD-I(Zhang et al., 2019b)	Homomorphic cryptography, threshold Paillier, one-way hash chain	Synthetic	Running time, communication overhead	Resistant to false data injections	Fully trusted authority, computational burden on users
RPTD-I(Zhang et al., 2019a)	Homomorphic cryptography, Paillier, one-way hash chain	Real world, simulated	Communication overhead, RMSE, running time	Fog computing	Fully trusted authority
PPTDS-I(Zhang et al., 2019c)	Data pertubation	Not specified	Running time	Low user effort	User data not encrypted when sent, fully trusted authority
EPTD-I(Zhang et al., 2020a)	Data perturbation	Data from custom application	MAE, RMSE, running time	Low user effort	Low level of security, fully trusted authority
SATE(Zhang et al., 2020b)	PCDD cryptosystem, data perturbation	Not specified	Communication overhead, running time	Low user effort	User data not encrypted when sent
DTD(Kang et al., 2023)	Majority Voting Inference, Trust Inference	SuavDS, AepDS	Specificity, Sensitivity, F1-score, Accuracy	Cost-effectiveness of worker recruitment	Not privacy preserving
UTD(Xiao and Wang, 2023)	Joint Maximum Likelihood Estimation	Education, SFV, Trec, Product, Sentiment	Precision, Acc., Recall, F1-score, MAE, RMSE	Includes many existing methods as special cases	Not privacy preserving
BLIND	Homomorphic cryptography, Paillier, K-Means	Real world, synthetic	RMSE, communication overhead, running time	Low user effort, open-source	Semi-honest third party

vacy. Many works investigate the techniques of PPC and homomorphic encryption, highlighting which elementary and complex operations can be carried out respecting data privacy constraints (Lagendijk et al., 2012; Rane and Boufounos, 2013). Table 1 summarizes the main characteristics of the most relevant systems proposed in the literature.

Among the first works that preserve privacy, PPTD (Miao et al., 2019) leverages homomorphic encryption techniques, i.e., the Threshold Paillier cryptosystem, to implement the iterative algorithm of CRH. However, the computation of truth values is not performed offline, and instead the system requires a direct involvement of users in the decryption of intermediate aggregated data with a consequent considerable computational and communication effort for users. In order to overcome these limitations, the same authors proposed two lightweight privacy-preserving truth discovery frameworks, L-PPTD and L^2-PPTD (Miao et al., 2017).

Both exploit two non-colluding cloud platforms and an additively homomorphic cryptosystem to assign weights to users' reports. However, the former relies on users to compute these weights in an encrypted manner, thus imposing a heavy computational burden on them, while the latter estimates weights in plaintext, putting users' privacy at risk. Conversely, in our solution, users only participate in the initial phase of the protocol, encrypting private values for each task and sending them to the server, thus ensuring their privacy as well as requiring low computational effort on their side. Xu et al. (2017) propose EPTD, a system that performs better than PPTD, but at the price of using a symmetric key shared between all users of the system, which makes it particularly vulnerable to malicious users and not feasible in practice.

Zheng et al. (2017) use the two-server model to create truth discovery systems that respect users' privacy. In the two-server model, one server collects and manages all user data in encrypted form, while the second one supports all those steps where decryption of intermediate aggregate data is required. Zheng et al. (2018) propose a privacy-aware truth discovery system that is designed specifically for the crowdensing application scenario, decreasing bandwidth and computational resources required from users.

Recently, Zhang et al. proposed several systems that implement the same CRH algorithm in a privacy preserving fashion and which differ for the application context, such as LPTD-I for cognitive Internet of Things (CIoT) (Zhang et al., 2019b), and RPTD-I for vehicular ad hoc network (VANET) (Zhang et al., 2019a).

Two alternatives proposed by the same authors that do not use homomorphic cryptosystems, but exploit data perturbation techniques, are EPTD-I (Zhang et al., 2020a) and PPTDS-I (Zhang et al., 2019c). A novel solution called SATE, presented in (Zhang et al., 2020b), solves the same problem by mixing homomorphic encryption and data pertubation. In recent years, several truth discovery methods have been proposed to infer true claims from multiple conflicting sources. (Xiao and Wang, 2023) provides a unified perspective on the existing methods and proposes a new algorithm. (Kang et al., 2023) proposes new techniques to improve the quality of collected data, including reducing bias and recruitment costs, and increasing accuracy and cost-effectiveness, which may have significant implications for the Industrial Internet of Things. However, neither of these solutions offers privacy guarantees.

Table 2 shows the comparison among the previous works from six points of view, including the techniques exploited, the dataset used in the experiments, the main measures used to evaluate results, the main advantages and disadvantages of each system. Analyzing Table 2, we can gain several insights on related work and their limitations compared to our approach. Many systems, such as CRH, DTD, and UTD, lack privacy-preserving measures altogether. In contrast, BLIND integrates the use of homomorphic cryptography, which guarantees user privacy to a significant extent. Some methods, such as PPTD and LPTD-I, impose a high computational burden on users. The latter also requires a fully trusted authority, which may not always be feasible. BLIND manages to reduce user effort, which is an important advantage for real-world application. Many existing systems, including LPTD-I, RPTD-I, PPTDS-I, and EPTD-I, rely on a fully trusted authority. This assumption can limit the applicability of these systems, since an authority like that may not always be available. In contrast, BLIND only assumes a semi-honest third party, making it more flexible and realistic for many use cases. Moreover, BLIND stands out as the only system listed that is open-source, which can facilitate its adoption and adaptation by other researchers and developers. Overall, BLIND presents a significant step forward in terms of privacy protection and real-world applicability, by reducing user effort. It employs a unique combination of techniques and offers the additional advantage of being open source.

Unlike many previous works, which exploit CRH for truth discovery, we propose an alternative model of QoI optimization, based on the clustering of user observations and the subsequent removal of outliers. A few works deal with clustering in a privacy preserving manner using K-Means (Mohassel et al., 2019; Yuan and Tian, 2017). In (Yuan and Tian, 2017), a single user holds all the data in plaintext and relies on a server for clustering in a privacy preserving manner. Such a system, however, is not suitable in a scenario where two or more users want to keep their data private. In (Jäschke and Armknecht, 2018), K-Means is used to clustering data from multiple users, but it is implemented using a fully homomorphic scheme, which makes it unusable in practice, resulting in run times of up to hundreds of days. Experimental results extensively show that BLIND outperforms its competitors while mitigating different security attacks.

3. Problem Formulation and System Model

The goal of a truth discovery system is to identify events when multiple data sources provide different information about one or more phenomena. We want to achieve this goal in a privacy-preserving fashion so that no one can exploit user information for malicious purposes that harm the interests of the participants. We therefore consider a cloud service provider (SP), which collects information from users in a privacy-preserving manner, without actually being able to view its content. Despite these limitations, the SP can aggregate the multitude of values coming from users, excluding false and noisy information, and finally obtain the truth value regarding a certain fact.

In BLIND, we assume the SP is as an honest-but-curious (semi-honest) party, which follows the protocol correctly, but tries to get as much information about the users as possible from the data received. This seems a rather weak adversarial model but it is adequate for a multitude of scenarios (Lagendijk et al., 2012; Shen et al., 2017). In addition, Goldreich et al. (1987) demonstrated how a protocol that computes a privacy-preserving function in the semi-honest model can be automatically converted into an equivalent protocol that maintains its privacy properties even in the malicious model.

We consider a set of n users $U = \{u_1, u_2, \ldots, u_n\}$. This set includes users willing to share their knowledge (e.g., through a smartphone application) about events of interest for the SP, motivated by a reward or incentive mechanism that generally leads them to participate only if their privacy is preserved. In our model, users can give false or inaccurate answers to the SP in order to obtain more rewards.

We also consider a set of q tasks $T = \{t_1, t_2, \ldots, t_q\}$ assigned to each of the users. Each task represents an infor-

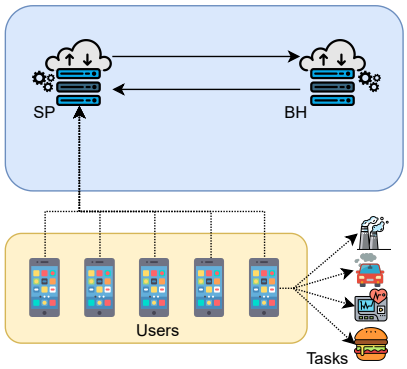


Figure 1: Entities involved in BLIND (Service Provider, Blind Helper, users).

Table 3: Notation used for the problem formulation.

Symbol	Description
SP	the service provider
BH	the blind helper
n	number of users
q	number of tasks
U	set of users
T	set of tasks
u_i	user i belonging to the set U
t_j	task j belonging to the set T
v_{ij}	reported value of user u_i for task t_j
V_i	reported values of user u_i for all the q tasks
U^*	set of unreliable users
u_i^*	i-th unreliable user belonging to the set U^*
V_i^*	noisy reported values of user u_i for all the q tasks
$\overline{v_j}$	estimated truth value for task t_j

mation of interest on which the system requires a response or opinion from users.

Therefore, the SP asks each user to send q v_{ij} answers values $V_i = \{v_{i1}, v_{i2}, \ldots, v_{iq}\}$, where v_{ij} is the answer of user u_i for task t_j. Although v_{ij} values individually may not put users' privacy at risk, they may reveal sensitive information about them when analyzed as a whole, thus compromising their privacy. It is essential that users never reveal this information in unencrypted form.

The SP should not receive plaintext data from users, nor should it have the tools (e.g. cryptographic keys) to decipher such data. At the same time, the SP has to calculate aggregate values from that secret data sent by users. To solve this problem we decided to take advantage of homomorphic cryptography that allows the SP to perform operations on encrypted data, and to include a third party that acts as a blind helper (BH), which securely holds the cryptographic keys useful for the execution of homomorphic protocols and *blindly* helps the SP with some computations, as will be explained in the following sections. The BH, like the SP, will be considered honest-but-curious, and it is assumed that the necessary legal and risk management measures are in place to ensure that BH and SP cannot collude to violate users' privacy. Figure 1 summarizes the scenario and the entities described above.

	t_1	t_2	t_3	t_4
u_1	v_{11}	v_{12}	v_{13}	v_{14}
u_2	v_{21}	v_{22}	v_{23}	v_{24}
u_3	v_{31}	v_{32}	v_{33}	v_{34}
u_4	v_{41}	v_{42}	v_{43}	v_{44}

Figure 2: Matrix showing how BLIND operates. Rows represent users, columns represent tasks.

The SP exploits the data collected on individual tasks without actually being able to read their content, discards noisy or false information and thus obtains a value as close as possible to the real one resulting from the aggregation of only good reports.

Figure 2 shows how BLIND operates. In this matrix, rows represent users, columns represent tasks and cells are values sent to the SP. We are interested in performing vertical aggregations to obtain the truth about a task (yellow selection in the picture). The process of discarding information sent by unreliable users is highlighted by the purple dashed selection in the figure.

Let us call V_i^* the noisy or false information provided by a user u_i^* belonging to the set of unreliable users U^*. Truth values $\overline{v_j}$ are calculated as a function of the values suggested by good users, discarding V_i^* values. Table 3 summarizes the notation used so far.

As we will see in the next section, homomorphic encryption can meet the needs of the scenario considered, allowing the SP to take into account only true information, excluding noisy or false reports without actually knowing their content, whilst preserving the confidentiality of the information sent by participating users.

4. Multi-Party Computation Background

In this section we will present some basic concepts and techniques of secure multi-party computation. In particular we will focus on a few homomorphic encryption properties provided by some important probabilistic encryption schemes (e.g., Paillier's scheme (Paillier, 1999)). Such schemes, unlike deterministic ones, produce different ciphertexts when encrypting the same plaintext. This property is obviously fundamental to maintain user privacy, especially when the number of possible answers is small (e.g. yes or no questions or binary values in general). We refer the reader to (Lagendijk et al., 2012; Fontaine and Galand, 2007) for an exhaustive documentation on all the properties we will present.

4.1. Homomorphic Additive Properties

Let $\mathcal{M}$ and $\mathcal{C}$ denote respectively the set of plaintexts and the set of ciphertexts. An encryption scheme is additively homomorphic if a relation of such form is satisfied:

$$E(m_1) \cdot_{\mathcal{C}} E(m_2) = E(m_1 +_{\mathcal{M}} m_2), \forall m_1, m_2 \in \mathcal{M} \quad (1)$$

where $\cdot_{\mathcal{C}}$ is a generic operation carried out on two encrypted messages which results in the encryption of the sum ($+_{\mathcal{M}}$) of the two plaintext messages. Paillier and Benaloh schemes are two of the most used asymmetric homomorphic schemes where such operation is the multiplication of two encrypted plaintexts. Note that both m_1 and m_2 must be encrypted with the same public key PK.

As consequence of Equation 1 we can obtain another relevant property. Any encrypted message $E(m)$ raised to the power of p results in the encryption $E(m \cdot p)$, as we can see in Equation 2:

$$E(m)^p = \prod_{i=1}^{p} E(m) = E\left(\sum_{i=1}^{p} m\right) = E(m \cdot_{\mathcal{M}} p), \forall m, p \in \mathcal{M} \tag{2}$$

Other properties may be easily derived from the two defined above. For example, the subtraction between encrypted messages $E(m_1)$ and $E(m_2)$ can be obtained as follows:

$$\begin{aligned} E(m_1 -_{\mathcal{M}} m_2) &= E(m_1) \cdot_{\mathcal{C}} E(-m_2) \\ &= E(m_1) \cdot_{\mathcal{C}} E(m_2)^{-1}, \forall m_1, m_2 \in \mathcal{M} \end{aligned} \tag{3}$$

The division between an encrypted message $E(m)$ and a plaintext scalar value p can be derived directly from Equation 2:

$$E\left(\frac{m}{p}\right) = E(m)^{\frac{1}{p}}, \forall m, p \in \mathcal{M} \tag{4}$$

Cryptosystems that allow the multiplication of two encrypted messages are called multiplicative homomorphic, while Paillier's cryptosystem is additively homomorphic.

In the following section we present the *blinding* technique, which allows more complex operations such as computing the square of an encrypted message or multiplying two encrypted messages. The disadvantage of this technique is that it requires a communication protocol between two parties, with an increase in computational complexity.

4.2. Blinding Technique to Multiply Encrypted Values

Suppose there are three entities, namely Alice, Bob, and Charlie, and consider the following scenario. Charlie wants Alice to perform computations on his secret message m, without Alice discovering the contents of m. For this purpose, Charlie encrypts m with the public key of a third party, Bob, and sends the resulting $E(m)$ to Alice.

If the computations are those described in the previous section, Alice can operate directly on the ciphertext as just discussed. To perform more complex operations (such as calculating the squared value), however, Alice needs to collaborate with Bob.

Indeed, Bob could easily help Alice by decrypting $E(m)$ and performing these operations, but Alice does not want him to know the value of the plaintext m (since that would violate Charlie's privacy), so she cannot send the message to Bob directly.

To solve this issue, Alice can alter the encrypted message by choosing a random value r, encrypting it with Bob's public key and multiplying the resulting $E(r)$ by $E(m)$. According to the first *additive property* (Equation 1), the following identity holds: $E(m) \cdot E(r) = E(m + r)$.

If r has the right random properties, Alice can safely send $E(m + r)$ to Bob. Bob can then decrypt the value, perform some operations, re-encrypt it and send the result back to Alice, without learning anything useful in the process. This technique, named *blinding*, can be very useful, for example, to calculate the square of an encrypted message $E(m)$ without knowing the key to decipher it.

Alice, who wants to know the value of $E(m^2)$, sends to Bob $E(m) \cdot E(r) = E(m + r) = E(z)$. Bob, knowing the private key, obtains z and calculates z^2. At this point, he encrypts z^2 with the public key and send $E(z^2)$ back to Alice, who can now obtain the value of $E(m^2)$ as follows:

$$\begin{aligned} E(m^2) &= E(z^2 - 2mr - r^2) \\ &= E(z^2) \cdot E(m)^{-2r} \cdot E(-r^2). \end{aligned} \tag{5}$$

By generalizing Equation 5, the blinding technique can also be used to compute the product between two messages that are only available in encrypted form. Note that such operation cannot be performed directly with homomorphically additive cryptosystems. Alice, who wants to obtain the encrypted value of the product of two messages, $E(m_1 \cdot m_2)$, sends to Bob the two values $E(m_1 + r_1) = E(z_1)$ and $E(m_2 + r_2) = E(z_2)$. Bob, knowing the private key, decrypts the values received, thus obtaining z_1 and z_2, and performs the product between the two values. Note that the blinding technique used prevents Bob from knowing the values of the messages m_1 and m_2, since both have been blinded with r_1 and r_2 values respectively. At this point, Bob encrypts the product with his public key and sends $E(z_1 \cdot z_2)$ to Alice, who can now compute the desired value as follows:

$$\begin{aligned} E(m_1 \cdot m_2) &= E(z_1 z_2 - m_1 r_2 - m_2 r_1 - r_1 r_2) \\ &= E(z_1 z_2) \cdot E(m_1)^{-r_2} \cdot E(m_2)^{-r_1} \cdot E(-r_1 r_2). \end{aligned} \tag{6}$$

4.3. Paillier Cryptosystem

A cryptosystem that has only the additive homomorphic property is called partially homomorphic. A cryptosystem that supports arbitrary calculation on encrypted texts is known as fully homomorphic. The existence of a fully homomorphic cryptosystem solves any problem of secure calculation from a theoretical point of view. Unfortunately, nowadays fully homomorphic cryptographic systems are mainly of theoretical interest and too inefficient to be used in practice. For this reason, we used an asymmetric partially homomorphic cryptosystem, namely the Paillier scheme.

The Paillier cryptosystem is a probabilistic asymmetric algorithm for public key cryptography and exhibits the additive homomorphic properties just described. The following steps are performed for the generation of the public and private key pair:

- Two large prime numbers of the same dimension p and q are chosen. Then, the following property is respected: $gcd(pq, (p-1)(q-1)) = 1$.
- Calculate the product $n = pq$, and $\lambda = lcm(p-1, q-1)$, where λ is the least common multiple of $p-1$ and $q-1$.
- Choose an integer random value g, with $g \in \mathcal{Z}_{n^2}$; in other words $g \leq n^2$. Make sure that for this value g, the modular multiplicative inverse μ exists, where μ is:

$$\mu = \left\lfloor \frac{(g^{\lambda} \mod n^2) - 1}{n} \right\rfloor^{-1} \mod n \qquad (7)$$

  The result of the division is different from zero only when g is at least greater than n.

The public key, PU_b, is given by the pair of values (n, g) while the private key, PR_b, is given by the pair (λ, μ).

Given a message m we want to encrypt with m positive integer less than n and a random integer r between 0 and n, the encrypted message c can be obtained from:

$$c = g^m r^n \mod n^2. \qquad (8)$$

The inverse operation, given an encrypted message c, allows us to recover the original message, as follows:

$$m = \frac{(c^{\lambda} \mod n^2) - 1}{n} \mu \mod n. \qquad (9)$$

Equations 8 and 9 show that each time the message m is encrypted, a different ciphertext c is obtained, even when using the same public key PU_b, since there is a random value r involved. It is also worth noting that r is not necessary for the decryption operation.

Since Pallier cryptosystem allows only the encryption of non-negative integers, and we are also interested in representing negative and fractional numbers, encoding is necessary prior to the encryption operations. In addition, the encoding used must maintain the homomorphic properties of Pallier cryptosystem.

5. The Privacy-Preserving Truth Discovery System

In this section we present BLIND, our privacy-preserving MCS system which exploits the multi-party computation techniques presented in Section 4. The goal of the service provider (SP) is to obtain aggregate results from users whilst maintaining the privacy of individual participants.

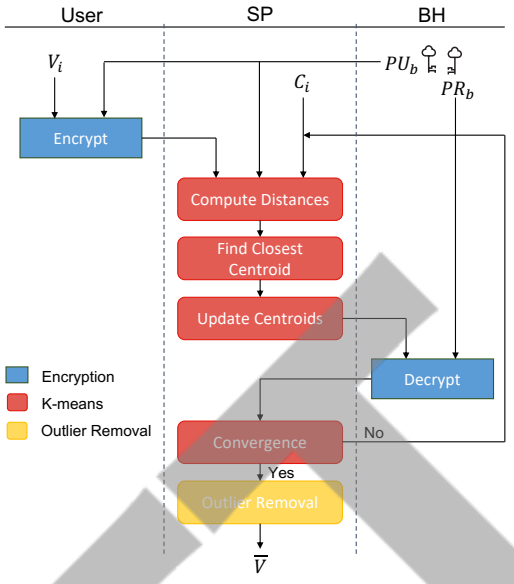


Figure 3: Architecture of the proposed system.

As shown in Figure 3, in order to aggregate data anonymously and, at the same time, improve the quality of information obtained by the SP, we propose a system that performs the following operations:

1. Encrypted data sent by users are grouped in clusters in a privacy-preserving way, through a modified version of the K-Means algorithm.
2. The result of the K-Means is used to discard outliers.
3. New centroids are calculated for each cluster, discarding the outliers.
4. The SP calculates a final truth value for each task, as a function of the centroids computed in step 3.

At all times during this process, values sent by users remain secret and known only to themselves. In addition, the SP does not need to know which cluster users belong to, or which users are considered outliers, in order to calculate the aggregate results of interest. Therefore, both are kept secret and are not known to either the SP or the users. Moreover, all messages are exchanged via a secure and authenticated communication channel, exploiting TLS. Ultimately, the SP learns the aggregate values for each task, either grouped by cluster (which may be of interest in certain usage scenarios) or as a single final value.

Given the notations introduced in Table 3 and Table 4, we can express the above requirements in a more formal manner. For all users $u_i \in U$, during the execution of protocol P, V_i is known only to u_i:

$$\forall u_i \in U : V_i \in \text{Knowledge}(u_i) \wedge V_i \notin \text{Knowledge}(SP) \wedge V_i \notin \text{Knowledge}(BH).$$

Table 4: Symbols used in the security protocol.

Symbol	Description
PU_b	public key of BH
PR_b	private key of BH
PU_s	public key of SP
PR_s	private key of SP
k	number of clusters
C_l	centroid of l-th cluster
D_i	vector of distances of V_i from the k centroids
d_{il}	distance of V_i from l-th centroid
A_i	k-dimensional indicator vector, with $a_{il} = 1$ if u_i belongs to the l-th cluster, or 0 otherwise
$E_b^+(\cdot)$	encrypted version of $(\cdot)$ by using PU_b
$E_s^+(\cdot)$	encrypted version of $(\cdot)$ by using PU_s
x'	blinded version of x using a random value r'
W	vector of s elements with increasing values $[w_1, w_2, ..., w_s]$, used for the *vectorization* technique
$\pi(\cdot)$	permutation of $(\cdot)$ by applying the permutation vector π

During the execution of protocol P, the association of user u_i to clusters (A_i) and the determination of outliers (set U^*) is kept private. Thus:

$$\begin{aligned}\forall u_i \in U : A_i \notin \text{Knowledge}(u_i) \wedge U^* \notin \text{Knowledge}(u_i) \wedge \\ A_i \notin \text{Knowledge}(SP) \wedge U^* \notin \text{Knowledge}(SP) \wedge \\ A_i \notin \text{Knowledge}(BH) \wedge U^* \notin \text{Knowledge}(BH).\end{aligned}$$

At the conclusion of protocol P, the SP obtains the estimated truth values $\overline{v_j}$, for each task t_j:

$$\forall t_j \in T : \overline{v_j} \in \text{Knowledge}(SP)$$

Depending on the service offered, the SP may then decide whether or not to share the final result with the users participating in the system.

5.1. Hypothesis and Limitations

The implementation of a privacy preserving mobile crowdsensing system using homomorphic encryption to ensure data privacy can be the right approach to protect the privacy of users and their sensitive data. However, there may be some limitations to consider, as with any security and privacy approach. The problem at hand is even more challenging than it may appear at first glance, because it is not possible to directly apply the techniques described in Section 4, which refer to a two-party computation scenario. If each participant used their own public-private key pair, it would not be possible to combine values from different users, as is necessary for K-Means. Naturally, the homomorphic properties of a cryptosystem are only guaranteed when the same key is used throughout the entire process. Ciphertexts obtained using different keys cannot be combined in any way. It is therefore necessary that all users encrypt their data with the same public key. The SP could generate the public-private key pair to use and distribute the public key to all participants, but this, although possible, would make the protocol more complex and leave most of the computationally heavy calculations to the users themselves.

Indeed, the SP could not perform these heavy calculations, as it owns the private key that would allow it to violate the secrecy of values sent by participants. Users would then have to perform many of the calculations normally carried out by the SP, which is not appropriate in mobile contexts, where users' devices have power and battery constraints. To avoid these problems, as discussed in Section 3, we propose to include another entity in the protocol, i.e., a third party that acts as a blind helper (BH). The BH will create a public-private key pair that will be used to encrypt users' secret values. Then, the BH will be asked by the SP to carry out some computations necessary to update the cluster centroids.

Extreme care will be taken to ensure that the BH can never decipher the secret information belonging to the users or to the SP. To this end, the SP will use the blinding techniques described in Section 4 when it is necessary to involve the BH in some calculations. Summarizing, thus, the BH will only have a supporting role, and will never learn any secret information about the users' data, their cluster membership, the outliers' identities, or the centroids.

Moreover, homomorphic encryption and the Paillier cryptosystem can be computationally intensive, requiring significant resources to compute and decrypt the data. Homomorphic encryption can also significantly increase the size of encrypted data compared to plaintext data, requiring more bandwidth and storage. This can be a problem in mobile devices where resources are limited.

With our approach, we shift the computational burden and the majority of message exchanges to the server side of the system, between the SP and the BH, in order to minimize the burden on users, who only contribute to the protocol in the initial phase. Scalability is another challenge for mobile crowdsensing systems in general. The use of homomorphic encryption can make the system less scalable, especially when it comes to a large number of participants and large amounts of data to be processed.

We will show in the experimental section that the approach proposed in this paper is scalable and suitable for the mobile crowdsensing context. Implementing a mobile crowdsensing system using homomorphic encryption and the Paillier cryptosystem requires advanced technical skills and attention to detail. An incorrect implementation could compromise the security of the system; any extensions or modifications require careful analysis.

Finally, the security of homomorphic encryption depends on the secure management of encryption keys. Loss or compromise of keys could lead to violations of privacy and data integrity. Despite these challenges, the use of homomorphic encryption is an important step in protecting privacy in mobile crowdsensing systems. However, it is crucial to carefully balance the benefits and limitations of these techniques to ensure the security and effectiveness of the overall system.

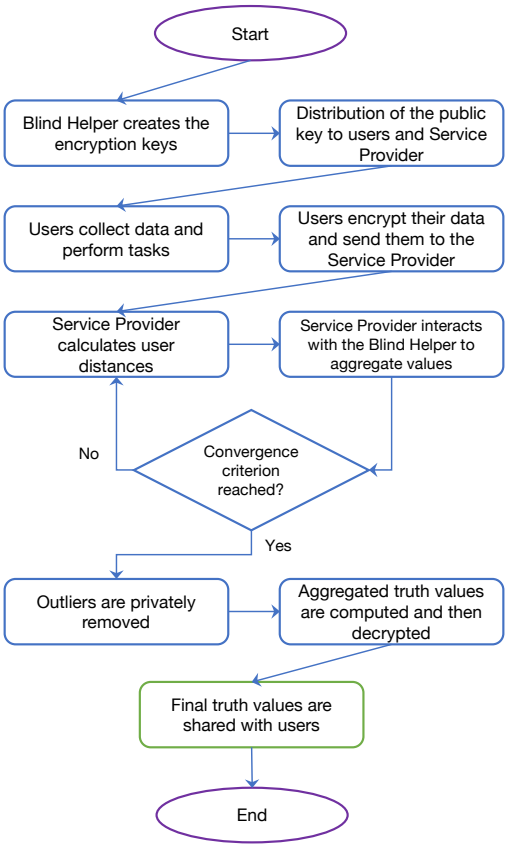


Figure 4: Flowchart of the operations performed by BLIND.

5.2. Privacy-Preserving K-Means Protocol

This section describes in detail the operations performed by BLIND in each phase of the protocol. Figure 4 is a flowchart that summarizes these operations and provides a visual guide to better understand how our system works.

At the start of the procedure, the BH generates the asymmetric key pair (PU_b and PR_b) needed for Paillier's encryption and shares the public key (PU_b) with both users and SP. In the following, we will denote with $E_b^+(x)$ values encrypted with the public key of the BH. Please note that all random values required for key generation and blinding techniques are obtained by exploiting a crypto-secure pseudo-random number generator (CSPRNG).

Users only participate in the initial phase of the protocol. In particular, the user u_i encrypts with PU_b his private values for each task, v_{ij}, and sends them to the SP. Users also send v_{ij}^2 to the SP, for each task, appropriately encrypted. These values will be used to calculate users' distances from cluster centroids, as will be explained in the following.

From this moment on, user participation is no longer required. This is important both to reduce the computational load on users' devices and to ensure that they cannot interfere with the execution of the protocol. Since we are considering q tasks in our application, the SP randomly generates k q-dimensional centroids, $C_1, C_2, \ldots, C_k$, which will be the starting values representing the k K-Means clusters. Squared Euclidean distance is used as error measure. The main task of the SP will be to update, iteratively, the values of the k centroids until the end of the K-Means algorithm (for example after a fixed number of iterations, or when the average distance from the nearest centroids converges).

The first step in each iteration is to calculate, for each user u_i, the encrypted version of $D_i = [d_{i1}, d_{i2}, \ldots, d_{ik}]$ distances between the V_i values and the k centroids. The SP can calculate the encrypted version of d_{il}, $E_b^+(d_{il})$, with the following formulas:

$$\begin{aligned} E_b^+(d_{il}) &= E_b^+\Big(\sum_{j=1}^{q}(v_{ij} - c_{lj})^2\Big) \\ &= E_b^+\Big(\sum_{j=1}^{q} v_{ij}^2\Big) \cdot E_b^+\Big(\sum_{j=1}^{q} v_{ij} \cdot (-2c_{lj})\Big) \cdot E_b^+\Big(\sum_{j=1}^{q} c_{lj}^2\Big) \\ &= \prod_{j=1}^{q} E_b^+(v_{ij}^2) \cdot \prod_{j=1}^{q} E_b^+(v_{ij})^{-2c_{lj}} \cdot \prod_{j=1}^{q} E_b^+(c_{lj}^2). \end{aligned} \tag{10}$$

Note that the SP knows the values of each c_{lj} and can encrypt them with PU_b, and that both $E_b^+(v_{ij})$ and $E_b^+(v_{ij}^2)$ are sent to it by user u_i. The SP then calculates Equation 10 and obtains $E_b^+(d_{il})$ for each user u_i and each centroid C_l.

Now the SP needs to calculate, for each user, the centroid with minimum distance, which is $\text{argmin}_l\, d_{il}$.

5.3. Privacy-Preserving argmin

Computing the encrypted version of $\text{argmin}_l\, d_{il}$ in a privacy preserving way is not a trivial task. The desired end result is the encrypted version of a k-dimensional indicator vector A_i, with $a_{il} = 1$ if u_i belongs to the l-th cluster, or 0 otherwise.

All this will be possible with the help of the BH. During the whole process, neither the SP nor the BH will have to acquire information about the distances of individual users from each cluster. Likewise, in the end, nobody will know which cluster each user belongs to. The only result will be the encrypted $E_b^+(A_i)$ vector for each user u_i, which represents $\text{argmin}_l\, d_{il}$.

To calculate the argmin, the SP will need to compare encrypted values (i.e., distances) to each other. However, this is not straightforward using an additive homomorphic cryptosystem such as Paillier. For this reason, we will first present a series of privacy-preserving protocols to perform these comparisons and subsequently we will describe the final protocol needed for calculating the actual argmin.

5.3.1. Secure Comparison between two Encrypted Values

In this section we will show how to compare two encrypted values in a privacy-preserving manner. We will proceed in steps, first introducing a protocol to compare an encrypted value, $E_b^+(x)$, with a plaintext one, y (e.g., a threshold) and then presenting another protocol to compare two encrypted values to each other. In both cases,

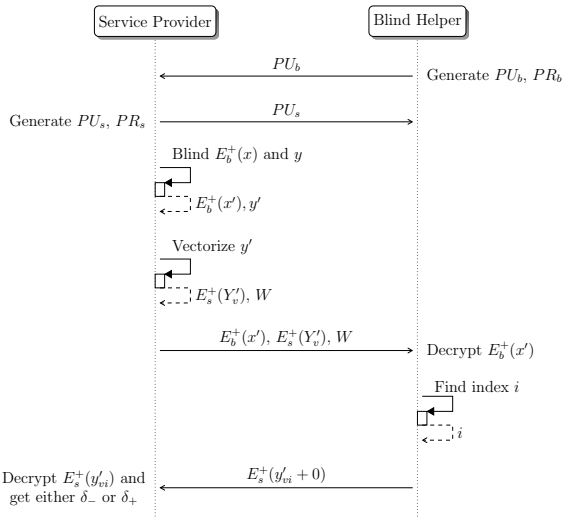


Figure 5: Sequence diagram describing the comparison between a ciphertext and a plaintext (Protocol 1).

the result of the comparison will be a plaintext boolean value (true or false).

However, we don't want the SP to learn the result of the comparison. We will thus show how to alter these protocols to obtain the result of the comparison in encrypted form (i.e., $E_b^+(0)$ or $E_b^+(1)$), so that the SP cannot decrypt it.

It is worth noticing that the Paillier cryptosystem operates on integers between 0 and n. As explained in Section 4.3, with an appropriate encoding it is also possible to represent and work with negative and fractional numbers.

In the following, for all proposed protocols, we will always assume that input values are properly encoded and that all results, including intermediate ones, are within the range of representable values. This is generally not a problem for reasonable key lengths (e.g., 1024 bits), but obviously longer keys can always be used if the application needs to represent unusually large values. A bigger key provides greater security, but it also results in longer execution times, as will be discussed in the experimental section.

Protocol 1. *Given a ciphertext $E_b^+(x)$ and a plaintext value y, the following protocol between the SP and the BH allows the SP to compare x and y without either party learning the value of x. The result of the comparison in plaintext will be only known by the SP.*

First, the SP generates a new asymmetric key pair (PU_s and PR_s) and shares the public key (PU_s) with the BH. In the following, we will denote with $E_s^+(x)$ values encrypted with the public key of the SP. To run the protocol, the SP must send $E_b^+(x)$ to the BH, which knows PR_b, and can thus decrypt the message. Since we do not want the BH to learn the value of x, the SP needs to *blind* this value before sending it, as explained in Section 4. To this end, the SP exploits a CSPRNG to choose a random value r' and computes the blinded value $E_b^+(x')$ as follows:

$$E_b^+(x') = E_b^+(x + r') = E_b^+(x) \cdot E_b^+(r'). \tag{11}$$

The SP then adds the same r' value to the plaintext value y, obtaining $y' = y + r'$. Since the same random value r' is added to both x and y, the order relationship between the two values remains unchanged, assuming that all values are within the representable range. At the same time, though, the SP can send the value $E_b^+(x')$ to the BH without revealing the value of x.

The problem we want to solve is a variation of the well known Yao's millionaire problem (Yao, 1982). To efficiently solve this problem, we heavily modified and adapted to our use case the *vectorization* technique proposed by Liu et al. (2017).

The SP creates a vector of z elements with increasing values, $W = [w_1, w_2, ..., w_z]$, with $w_1 \leq y' \leq w_z$. As will be discussed in the following, the number of elements of W and the granularity of the values $[w_1, w_2, ..., w_z]$ will determine a trade-off between computational efficiency and result accuracy.

The goal of the SP is to represent y' as a vector of z values, $Y'_v = [y'_{v1}, y'_{v2}, ..., y'_{vz}]$. The SP chooses two values, δ_- and δ_+, which will be used to fill Y'_v. More precisely, y'_{vi} will be equal to δ_- if $y' \leq w_i$, or δ_+ otherwise, $\forall i \in \{1, ..., z\}$. Note that, with the homomorphic encryption system used, repeatedly encrypting the same value (e.g., δ_- or δ_+) always generates different ciphertexts, as described in Section 4.

After that, the SP sends $E_b^+(x')$, $E_s^+(Y'_v)$, and W to the BH, which decrypts $E_b^+(x')$ and obtains x'. The BH finds the smallest index i for which $x' \leq w_i$, and returns

$$E_s^+(y'_{vi} + 0) = E_s^+(y'_{vi}) \cdot E_s^+(0) \tag{12}$$

to the SP. Finally, the SP decrypts $E_s^+(y'_{vi})$ and obtains y'_{vi}, which will be equal to δ_- if x is less than or equal to y, or δ_+ otherwise.

Figure 5 summarizes the protocol just described. Note that the BH cannot decrypt $E_s^+(y'_{vi})$, so it does not know the result of the comparison. Also note that the BH multiplies $E_s^+(y'_{vi})$ by $E_s^+(0)$ before returning it to the SP. This operation does not change the value of y'_{vi}, once decrypted (it is equivalent to adding 0). The effect of this *null sum*, however, is to change the ciphertext of $E_s^+(y'_{vi})$. If the BH did not do this, the SP could simply compare the returned ciphertext with the ciphertexts of the $E_s^+(Y'_v)$ elements and find out which is the smallest value of W that is greater than x.

A further consideration relates to the number of elements of W and the granularity of the values $[w_1, w_2, ..., w_z]$. The more fine-grained the W values are, in fact, the more accurate the comparison will be. At the same time, however, the number of encryption operations performed will be proportional to the number of elements of W. Note that the protocol may incur non-negligible

errors if the vectorization returns the same result for different values; this is the case if the difference between the two values is less than the gap between two consecutive elements of W. For example, if $W = [\ldots, 1.4, 1.5, 1.6, \ldots]$ and the two compared values are $x = 1.56$ and $y = 1.55$, Protocol 1 will erroneously return δ_-. Since the SP knows the range of values used in the application, it can always choose an appropriate granularity to avoid such errors. Moreover, the values in W do not need to be evenly distributed. Conversely, they can also be sparse, depending on the expected distribution of values to compare, using fine granularities for some ranges and coarse granularities for others.

Example. We now present a simple numerical example to clarify the comparison procedure described above. In this example, the SP wants to know if $E_b^+(x)$, with $x = 5.4$, is less than or equal to $y = 5.2$. Let us assume the SP chooses $\delta_- = 12.6$, $\delta_+ = -2.1$ and $r' = -3.1$. First, the SP blinds both $E_b^+(x)$ and y with r', obtaining $E_b^+(x') = E_b^+(2.3)$ and $y' = 2.1$. Then, the SP vectorizes y'. To do so, if the SP chooses $s = 7$ and W as follows:

$$B = [1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5],$$

the vectorized y' will be filled with δ_- and δ_+ accordingly:

$$\begin{aligned} Y'_v &= [\delta_-, \delta_-, \delta_-, \delta_+, \delta_+, \delta_+, \delta_+] \\ &= [12.6, 12.6, 12.6, -2.1, -2.1, -2.1, -2.1]. \end{aligned}$$

The SP then encrypts Y'_v with PU_s and sends it to the BH, together with $E_b^+(x')$ and the unencrypted W. The BH decrypts $E_b^+(x')$, obtaining $x' = 2.3$, and finds the lowest index i for which $x' \leq w_i$. If we consider indices starting from 1, then $i = 5$ in this case, because $w_6 = 2.3$ is equal to $x' = 2.3$.

The BH then returns $E_s^+(y'_{v6} + 0) = E_s^+(-2.1)$ to the SP. The SP decrypts it, obtaining $y'_{v6} = -2.1$, which is δ_+, and infers that x is greater than y, without learning anything about x.

The following protocol shows how to compute the comparison between two ciphertexts in a privacy-preserving manner.

Protocol 2. *Given two ciphertexts, $E_b^+(x_1)$ and $E_b^+(x_2)$, the following protocol between the SP and the BH allows the SP to compare x_1 and x_2 without either party learning the values of x_1 or x_2.*

The protocol proceeds as follows. First, the SP computes the following value:

$$E_b^+(x_1 - x_2) = \frac{E_b^+(x_1)}{E_b^+(x_2)} = \gamma. \tag{13}$$

Then, the SP can use Protocol 1 to compare γ with the plaintext value 0, obtaining the desired result.

Finally, as mentioned at the beginning of this section, we are interested in a variant of the problem where the SP does not get the result of the comparison in plaintext. Instead, we want the SP to get $E_b^+(0)$ or $E_b^+(1)$, so that it cannot decrypt it. To solve this new problem, we introduce a new protocol.

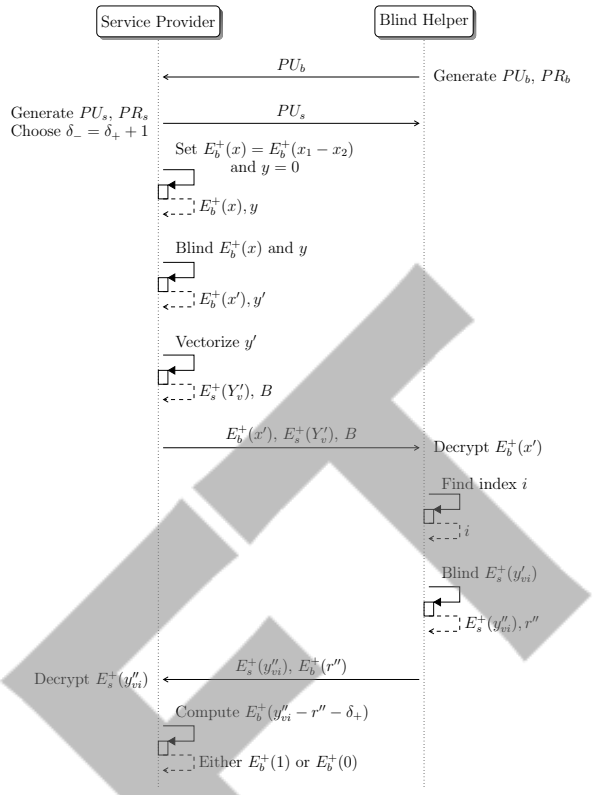


Figure 6: Sequence diagram describing the comparison between two ciphertexts values with encrypted result (Protocol 3).

Protocol 3. *Given two ciphertexts, $E_b^+(x_1)$ and $E_b^+(x_2)$, the following protocol between the SP and the BH allows the SP to compare x_1 and x_2 without either party learning the values of x_1 or x_2, or the result of the comparison. At the end of the protocol, the SP will get $E_b^+(1)$ if x_1 is less or equal than x_2, or $E_b^+(0)$ otherwise.*

First, the SP chooses δ_- and δ_+ such that $\delta_- = \delta_+ + 1$. Then, we proceed as in Protocol 2 until the BH is supposed to return $E_s^+(y'_{vi})$ to the SP (Equation 12). Instead of returning the value directly, the BH chooses a random value r'' and *blinds* the value as follows:

$$E_s^+(y''_{vi}) = E_s^+(y'_{vi} + r'') = E_s^+(y'_{vi}) \cdot E_s^+(r''). \tag{14}$$

Then, the BH encrypts r'' with PU_b and sends both $E_s^+(y''_{vi})$ and $E_b^+(r'')$ to the SP. The SP decrypts $E_s^+(y''_{vi})$, obtaining y''_{vi}. Finally, the SP encrypts both y''_{vi} and δ_+ with PU_b and computes the encrypted result of the comparison between x_1 and x_2 as follows:

$$E_b^+(y''_{vi} - r'' - \delta_+) = \frac{E_b^+(y''_{vi})}{E_b^+(r'') \cdot E_b^+(\delta_+)}. \tag{15}$$

$E_b^+(y''_{vi} - r'')$ will be either $E_b^+(\delta_-)$ or $E_b^+(\delta_+)$. Since the SP chose δ_- and δ_+ such that $\delta_- = \delta_+ + 1$, subtracting δ_+

from $E_b^+(y''_{vi} - r'')$ yields $E_b^+(1)$ if x_1 is less or equal than x_2, or $E_b^+(0)$ otherwise. This concludes the protocol.

Again, note that repeated encryptions of the same values (0 or 1) always yield different ciphertexts. Figure 6 shows the whole protocol.

In summary, using Protocol 3 the SP is able to compare two encrypted values, $E_b^+(x_1)$ and $E_b^+(x_2)$, obtaining the result in encrypted form, without either the SP nor the BH learning anything about x_1 and x_2.

5.3.2. Privacy-Preserving argmin Protocol

Now that we know how to compare encrypted values, we can present a novel protocol to calculate the encypted version of $\text{argmin}_l\, d_{il}$ in a privacy preserving way. The goal, as written above, will be to obtain, for each user u_i, the encrypted k-dimensional indicator vector $A_i = [a_{i1}, a_{i2}, ..., a_{ik}]$, with $a_{il} = 1$ if user u_i belongs to the l-th cluster (i.e., if u_i's values, V_i, are closest to the l-th centroid, C_l), or 0 otherwise.

Intuitively we could use the comparison between two encrypted values, as described in the previous section, to calculate the argmin directly, by securely comparing each encrypted distance with all the others. However, this is inefficient because it would require $k \cdot (k-1)$ secure comparisons, where k is the number of clusters. Given that comparisons are very expensive operations, computationally-wise, this might be a problem.

For this reason, we propose to exploit the comparison procedure in which the SP learns the result of the comparison in plaintext (true or false) to efficiently calculate the $E_b^+(A_i)$ vector. Obviously, this procedure cannot be applied directly, because otherwise the SP would learn the index of the cluster closest to each user (plus other information from intermediate calculations).

The goal is then to permute the distance vector D_i before computing the A_i indicator vector. This can be done with the help of the BH, which causes other problems, since the BH knows the private key to decipher $E_b^+(D_i)$ and cannot handle this data directly. We introduce a novel protocol exploiting the blinding and permutation techniques described by Rane and Boufounos (2013).

Protocol 4. *Given the encrypted distances $E_b^+(d_{il})$ of user i with each cluster centroid l, the following protocol between the SP and the BH uses $(k-1)$ secure comparisons to compute the encrypted version of* $\text{argmin}_l\, d_{il}$ *without either party learning the value of any of the d_{il}, or the plaintext version of the* argmin*, or any other private information about users.*

First, the SP creates a k-dimensional vector $R' = [r'_1, r'_2, \ldots, r'_k]$ of random values, which is used to blind the k values of the vector D_i as follows:

$$E_b^+(D'_i) = E_b^+(D_i) \cdot E_b^+(-R') = E_b^+(D_i - R'), \qquad (16)$$

where $E_b^+(-R')$ is multiplied by $E_b^+(D_i)$ element-wise to obtain $E_b^+(D_i - R')$, as described in Section 4.2. Note that

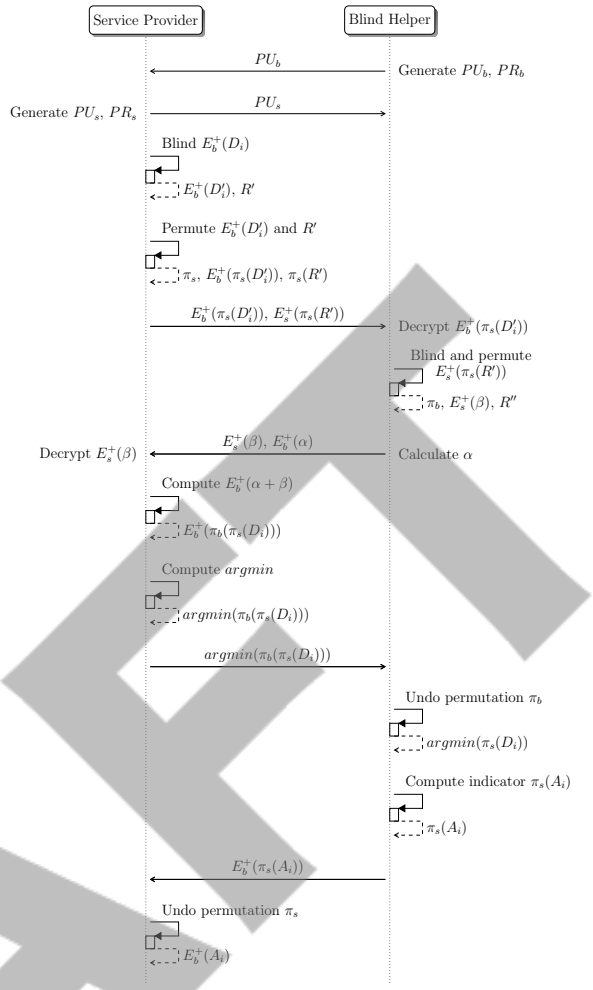


Figure 7: Sequence diagram describing the protocol to compute the encrypted version of argmin (Protocol 4).

$-R'$ is used in the product instead of R' to streamline subsequent calculations. Then, the SP generates a random permutation vector, π_s, and permutes both $E_b^+(D'_i)$ and R', obtaining, respectively, $\pi_s(E_b^+(D'_i))$ and $\pi_s(R')$. Since the encryption of a vector is element-wise, it is always true that $\pi(E(X)) = E(\pi(X))$ for every permutation π and vector X. In the following, we will denote the encryption as the outermost operation, for clarity of notation.

At this point, the SP encrypts $\pi_s(R')$ with its public key, and sends both $E_b^+(\pi_s(D'_i))$ and $E_s^+(\pi_s(R'))$ to the BH.

The BH decrypts $E_b^+(\pi_s(D'_i))$ and gets $\pi_s(D'_i)$. Note that the BH cannot infer any useful information from $\pi_s(D'_i)$, because its values are both blinded and permuted.

The BH creates a k-dimensional vector $R'' = [r''_1, r''_2, \ldots, r''_k]$ of random values and a random permutation vector, π_b. Then, the BH uses both these vectors to blind and permute $E_s^+(\pi_s(R'))$, as follows:

$$E_s^+(\beta) = E_s^+(\pi_b(\pi_s(R') - R'')), \qquad (17)$$

where R'' is subtracted from $\pi_s(R')$ element-wise. The BH

also computes the vector α as follows:

$$\alpha = \pi_b(\pi_s(D'_i) + R''), \tag{18}$$

where, again, R'' is added to $\pi_s(D'_i)$ element-wise. Finally, the BH encrypts α with its public key, and sends both $E_s^+(\beta)$ and $E_b^+(\alpha)$ to the SP.

The SP decrypts $E_s^+(\beta)$, obtaining $\beta = \pi_b(\pi_s(R') - R'')$. Now the SP possesses both $E_b^+(\alpha)$ and β. Note that $\alpha + \beta$ is a very interesting vector, as shown by the following formulas:

$$\begin{aligned} \alpha + \beta &= \pi_b(\pi_s(D'_i) + R'') + \pi_b(\pi_s(R') - R'') \\ &= \pi_b(\pi_s(D'_i) + R'' + \pi_s(R') - R'') \\ &= \pi_b(\pi_s(D'_i + R')) \\ &= \pi_b(\pi_s(D_i)), \end{aligned} \tag{19}$$

where the last step of Equation 19 is obtained by using Equation 16. The SP cannot compute $\alpha + \beta$ directly, because it only knows $E_b^+(\alpha)$ and β. It can, however, encrypt β with PU_b, and sum it to $E_b^+(\alpha)$ element-wise, obtaining $E_b^+(\alpha+\beta)$. From Equation 19, we can derive the following formula:

$$E_b^+(\alpha + \beta) = E_b^+(\pi_b(\pi_s(D_i))), \tag{20}$$

where $E_b^+(\pi_b(\pi_s(D_i)))$ is the encrypted version of the D_i vector, permuted both by the SP and the BH. At this point, the SP can trivially calculate the argmin of $E_b^+(\pi_b(\pi_s(D_i)))$, comparing the elements of the vector with the current (encrypted) minimum, keeping track of the corresponding index. Since the vector $E_b^+(\pi_b(\pi_s(D_i)))$ is permuted, the SP can repeatedly use the comparison procedure that returns a boolean value in plaintext (Protocol 2), without learning anything about the argmin of the original vector $E_b^+(D_i)$. Indeed, the SP can simply compute the minimum encrypted value in $E_b^+(\pi_b(\pi_s(D_i)))$ with $(k-1)$ comparisons, as required, and obtain the corresponding index, which is the argmin of $E_b^+(\pi_b(\pi_s(D_i)))$.

The SP then sends the argmin of the doubly permuted vector to the BH. The BH uses π_b, which it knows, to undo one of the two permutations, and to get the argmin of the $\pi_s(D_i)$ vector.

Note that, again, the BH does not get any new information, since it only learns the argmin of the permuted vector. The BH uses this value to create the indicator vector $\pi_s(A_i)$, where all elements are equal to 0, except for the one corresponding to the argmin of $\pi_s(D_i)$, which is equal to 1.

Then, the BH encrypts $\pi_s(A_i)$ with PU_b, and sends $E_b^+(\pi_s(A_i))$ to the SP. Finally, the SP uses π_s, which it knows, to undo the last permutation, obtaining $E_b^+(A_i)$ as desired. Figure 7 shows the whole protocol.

5.4. Privacy-Preserving centroids update

The next step is updating the C_l centroids, for each $l = 1, 2, \ldots, k$, computing the sum:

$$E_b^+(C_l) = E_b^+\left(\frac{1}{|U^l|}\sum_{i=1}^{n} V_i \cdot a_{il}\right), \tag{21}$$

where U^l is the set of users belonging to the l-th cluster, and $|U^l|$ is its cardinality. To calculate the terms of the sum in Equation 21, the SP needs to obtain, for each task j, $E_b^+(v_{ij} \cdot a_{il})$, where v_{ij} is an element of vector V_i, and a_{il} is a scalar value. Both V_i and a_{il} are known to the SP only in encrypted form. This operation is possible by exploiting a protocol that is executed with the help of the BH, using blinding techniques to keep secret values from being revealed, as explained in Section 4. The sum in Equation 21 can be easily calculated using the additive homomorphic property, as follows:

$$E_b^+\left(\sum_{i=1}^{n} V_i \cdot a_{il}\right) = \prod_{i=1}^{n} E_b^+(V_i \cdot a_{il}). \tag{22}$$

In a similar way, the factor $1/|U^l|$ in Equation 21 can be calculated by summing up the encrypted values a_{il} for each u_i and then it can be multiplied using blinding techniques.

At the end of each iteration, then, the SP gets $E_b^+(C_l)$ for $l = 1, 2, \ldots k$. By exploiting, once again, blinding techniques with the BH, the SP decrypts these values and learns the updated unencrypted centroids, which will be used in the next iteration to calculate the new distances from users' values. Note, once again, that the only values known by the SP are the centroids (aggregated values), and that nobody knows which clusters each user belongs to (not even the users themselves). These properties are maintained until the last iteration, when the K-Means ends.

5.5. Privacy-Preserving Outlier Removal

In order to remove the outliers in a privacy-preserving way, and thus improve the accuracy of our truth discovery system, as will be shown in the experimental section, the SP must take an additional step, which is a modified version of one of the K-Means iterations. The SP recalculates the distance of the users' values to the centroids, and identifies, for each user, the minimum distance $E_b^+(d_{il})$ and the encrypted vector $E_b^+(A_i)$, as explained in the previous section.

At this point the SP compares $E_b^+(d_{il})$ with a plaintext threshold. The comparison can be done in a privacy-preserving manner, as in Protocol 3.

The output will still be an encrypted value, $E_b^+(o_i)$. In particular, o_i will be equal to 0 if u_i is considered an outlier, or 1 otherwise. Now the SP calculates, with blinding techniques, the vector $E_b^+(\tilde{A}_i)$ for each user, as follows:

$$E_b^+(\tilde{A}_i) = E_b^+(A_i \cdot o_i). \tag{23}$$

This new vector $\tilde{A}_i$ takes into account, anonymously, whether a user has been classified as an outlier or not. $\tilde{A}_i$ is then used instead of A_i to update the centroids one last time, as described above. As a result, the SP gets the k cluster centroids without outliers, in a privacy-preserving way.

In the end, the SP can obtain the final truth for each task j by calculating the following weighted average:

$$E_b^+(\overline{v_j}) = E_b^+\left(\frac{1}{|U|}\cdot\sum_{l=1}^{k}|U^l|\cdot c_{lj}\right) = \prod_{l=1}^{k}E_b^+(|U^l|)^{\frac{c_{lj}}{|U|}}. \quad (24)$$

Considering that the SP knows both $|U|$ and all c_{lj}, Equation 24 is easy to calculate. Finally, the BH helps the SP to decipher $\overline{v_j}$, using blinding techniques to prevent the BH from learning unnecessary information.

Depending on the application scenario, alternative functions to obtain a single truth value might be preferable. For example, the SP could easily choose the *best* centroid, i.e., the one with the most users, by exploiting the encrypted information it has available.

6. Security Analysis

In this section we present the security analysis of BLIND by following an approach similar to Miao et al. (2017), demonstrating that the SP is able to optimize QoI by clustering user observations, while not revealing their private data to any of the parties involved in the protocol.

Suppose there are $n \geq 2$ users and $q \geq 1$ tasks, and that users have sent their encrypted values to the SP. If both SP and BH are honest-but-curious and are not colluding with each other, then values sent by users remain secret and known only to themselves at all times during the QoI optimization and thereafter. Furthermore, none of the parties involved will learn each user's cluster membership or the identity of the outliers.

To prove the security of BLIND, we need to show that none of the parties involved, i.e., users, SP, and BH, learn any secret information. Users do not receive any information about other participants, neither in plaintext nor encrypted form. Indeed, their only involvement occurs at the beginning of the protocol, when participants send their observations. Please remember that all messages are exchanged via a secure communication channel. Nevertheless, even if users were able to intercept some of the messages, they would not be able to decrypt them because they do not know the private key PR_b.

Now, we need to show that SP and BH also do not gain access to confidential information and cannot infer anything even in the intermediate protocol steps. Recall that SP and BH are honest-but-curious, which means that they follow the protocol, but may try to infer extra information. We will proceed with a step-by-step analysis of the data held by SP and BH.

At the beginning of the protocol, the SP knows $E_b^+(v_{ij})$ and $E_b^+(v_{ij}^2)$ for all users i and all tasks j. Since the SP does not know the related private key, however, it cannot determine the corresponding plaintext values. Remember that repeatedly encrypting the same value always generates different ciphertexts, as described in Section 4, thus the SP cannot infer whether two users have submitted the same values for a given task.

The SP can then calculate $E_b^+(d_{il})$ for each user u_i and each centroid C_l, i.e., the encrypted distances between users and centroids, by applying Equation 10. This is done by exploiting the homomorphic properties of the Paillier cryptosystem, without revealing the plaintext versions of user values and distances and without involving the BH.

Then, the SP has to find, for each user, the nearest centroid by computing the encrypted version of $\text{argmin}_l\, d_{il}$. As shown in Protocol 4, the SP can compute these values working together with the BH, without either party learning (i) the value of any of the d_{il}, (ii) the plaintext version of the argmin, or (iii) any other private information about users.

The next step entails the SP updating the centroids, by using Equations 21 and 22. This operation is possible by exploiting protocols executed with the help of the BH, using blinding techniques to keep secret values from being revealed, as demonstrated in Section 5.4.

At the end of each iteration, then, the SP gets the encrypted centroids $E_b^+(C_l)$. Using, once again, blinding techniques with the BH, the SP decrypts these values and learns the updated unencrypted centroids, which will be used in the next iteration to calculate the new distances from users' values. In summary, the only plaintext values known by the SP are the centroids. Neither the SP nor the BH learn each user's cluster membership. These properties are maintained in all iterations until the end of the K-Means protocol.

Then, in the outlier removal phase, the SP compares each encrypted distance $E_b^+(d_{il})$ with a threshold, to determine whether or not to include each user's values in the final truth computation. Using Protocol 3, the SP is able to perform these comparisons in a privacy-preserving manner, obtaining the result in encrypted form, as a vector that takes into account, anonymously, whether a user is an outlier or not. Finally, the SP can obtain the estimated truth values for each task by computing Equation 24 with the help of the BH, once again exploiting blinding techniques to avoid revealing secret values.

7. Experimental Results

In this section we will analyze the performance of the proposed truth discovery system from different points of view. In the first part we will discuss the accuracy of our approach, comparing BLIND with existing relevant systems. All the comparisons have been performed on a real world dataset that collects opinions sent by users in ten years, from 2008 to 2017. In order to thoroughly test the resilience of all compared systems against attacks of various kinds that aim to decrease the quality of information, we will present several attack scenarios that model realistic

situations. In the second part we will compare the communication overhead of BLIND with other relevant work. Finally, in the third part of this experimental section we will show the scalability of our approach by varying different parameters, such as number of users, number of tasks, number of clusters and key length. In order to test the scalability of the system over an extensive range of parameters, we used synthetic data produced specifically to reflect real-world scenarios. In addition to testing the scalability of the system on the server side, we also conducted a series of experiments to verify scalability from the perspective of participants using their mobile devices. Considering the mobile crowdsensing application scenario, this final set of tests is extremely important. In fact, while the servers can be scaled both horizontally and vertically, it is critical that the system be usable by participants with older devices. Thus, these latest tests show how BLIND can actually be used in real-world applications.

All of our experiments were carried out using a VM equipped with 8 GB RAM and 8 cores Intel(R) Xeon(R) CPU E5-2643.

7.1. QoI Optimization Comparison

In this section we evaluate the accuracy of BLIND, testing it with a real world dataset to verify the effectiveness of our approach. For comparison reasons, we implemented CRH (Li et al., 2016), PPTD (Miao et al., 2019), DTD (Kang et al., 2023) and UTD (Xiao and Wang, 2023). To the best of our knowledge, CRH and PPTD are considered among the most relevant work in the field, since they inspired many other approaches. CRH is not a privacy-preserving system, but it is still a benchmark for truth discovery accuracy.

Many of other systems inspired by CRH, including PPTD, differ primarily in aspects related to maintaining users' privacy, while the formulas used for truth discovery are very similar to each other (or, in some cases, identical). In particular, L-PPTD (Miao et al., 2017), EPTD (Xu et al., 2017), LPTD-I (Zhang et al., 2019b), and RPTD-I (Zhang et al., 2019a) obtain the same QoI results as PPTD. For presentation reasons, in the following graphs we indicate only PPTD as the representative system of all this category.

Although CRH, in its basic version, would be equivalent to PPTD, the authors of the former point out that their system is susceptible to the presence of outliers. To mitigate the impact of outliers, a variant of CRH uses alternative formulas that exploit the normalized absolute deviation as loss function and selects the truth value based on the weighted median of values. In the experiments, we present this outlier-resistant variant of CRH (henceforth referred to simply as CRH).

DTD and UTD, like CRH, are not privacy-preserving systems. Evaluating the accuracy of a privacy-preserving TD system like BLIND against truth-discovery systems that don't prioritize privacy is both important and challenging. This is largely because preserving privacy often involves trade-offs, and systems that focus primarily on truth discovery may use sophisticated techniques that may produce superior results, albeit without preserving privacy.

7.1.1. Methodology

To carry out our experiments, we used a real world dataset from the UCSD Book Graph (Wan and McAuley, 2018) project, collected by the UC San Diego University. The data was extracted from the popular book review site Goodreads in 2017 and then updated in 2019. The full dataset contains information on approximately 2.3 million books, 876 thousand users and over 100 million ratings expressed by users on books. Each rating submitted by a user is a value ranging from 1 (lowest) to 5 (highest) stars.

Given the large amount of total data, the full dataset is also divided into sub-datasets based on the genre of the books. Some of these sub-datasets are denser (i.e., the number of ratings per book is higher) than others. In particular, we wanted to select the largest possible set of users who all rated the same books. Doing this, however, is more difficult than it might seem. We can represent this problem in the form of a matrix, where the rows denote users and the columns denote books. A value of 1 in cell (i, j) indicates that the i-th user rated the j-th book, while a value of 0 indicates that the book was not rated by that user. However, selecting the largest non-contiguous rectangular sub-matrix consisting only of 1-entries is an *NP-complete* problem called *maximum edge biclique* (Peeters, 2003). For cases with a lot of data it is not feasible to find an exact solution.

To avoid this problem, we adopted a simple heuristic, by first selecting the n books with the highest number of ratings, and including in the test data only the users who have rated all of them. Doing so, we obtained a clique of 98 users and 10 books, which serves as the basis for our experiments. Our ground truth consists of the average ratings of the books in the full dataset, which takes into account years of ratings submitted by all Goodreads users.

In order to fully test the capabilities of the compared systems in improving the quality of information, even in the presence of noisy data or malicious users, we also considered different variants of the dataset, envisaging multiple attack patterns. Specifically, four different types of attacks are considered in the experiments we present:

- *Random* attack, where attackers (which in this case could also be regular "lazy" users) send random ratings for all books.

- *Slandering against one target*, in which each attacker decides independently a target and tries to penalize it, reducing its average rating; to implement this type of attack using as much real data as possible, we have considered real users who rated $n-1$ books, and set to 1 (minimum score) their rating for the remaining book.

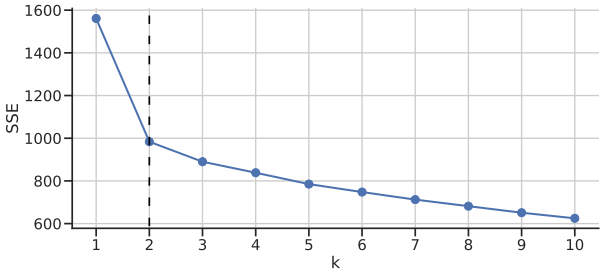


Figure 8: Results of the elbow method.

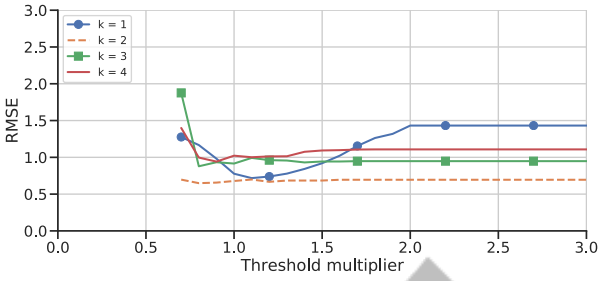


Figure 9: RMSE obtained by different variants of BLIND when varying the threshold multiplier.

- *Slandering against two targets*, similar to the previous case, except that the targets are two different books, selected independently by each attacker.

- *Orchestrated slandering*, in which the attackers collude to lower the ratings of the j-th book in a coordinated way; to implement this type of attack we selected real users who rated all books except the j-th, setting to 1 their rating for the book targeted by the attack.

We have chosen the RMSE (Root Mean Squared Error) as the main metric to measure the accuracy of the systems. All experiments have been repeated 1000 times and we present the average of the results obtained. We also conducted analysis of variance tests for all considered scenarios to show that there is a statistically significant difference among the performance of the various truth discovery systems.

7.1.2. QoI Optimization Results

Before comparing with other systems, we present an analysis of the two main parameters of BLIND in order to study how they affect performance. The parameters we consider are the number of clusters, k, and the threshold to discard outliers.

As regards k, we have employed the traditional approach used for K-Means, called *elbow* method. The idea is to run K-Means with different values of k and calculate the sum of squared errors (SSE), as shown in Figure 8. The point in this curve where the slope changes drastically, which looks like an *elbow*, indicates the best value of k to use, so that the number of clusters is not too large. Figure 8 shows the result of these experiments, highlighting how 2 is the value of k to use in our case.

Another set of experiments focused on the choice of the threshold to use for discarding outliers. We defined the threshold as a multiplier in relation to the average distance of user values from their respective centroids. A threshold of 2, for example, discards those values which have a distance more than double the average.

As expected, the performance of BLIND as k varies is also affected by the number and type of attackers. In general, we have observed that a small number of clusters is better suited to cases where there are few attackers, while using more clusters results in a system that performs better in the presence of a large number of attackers and less well if there are few of them. Our goal in this set of experiments is to estimate reasonable values for the parameters, which are suitable for different situations that could happen in a real context. For the sake of this experiment, then, we selected a medium-low number of attackers (30) and the simplest type of attack, as described earlier (*random* attack).

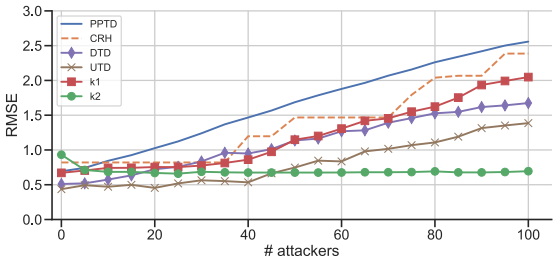
Figure 9 presents a summary of the experiments we conducted. This figure shows how variations in both the number of clusters k and the threshold level impact the RMSE values obtained. Systems demonstrating lower RMSE values are indicative of superior performance due to fewer inaccuracies. A higher threshold implies that more values, even those that might be on the borderline of being outliers, are included in the calculations. On the flip side, when the threshold is reduced, the systems become more stringent, thereby discarding a larger amount of outliers.

Our results correlate well with the elbow method's findings, where the system with $k = 2$ clusters consistently yields the most effective results, proving its robustness to various threshold settings.

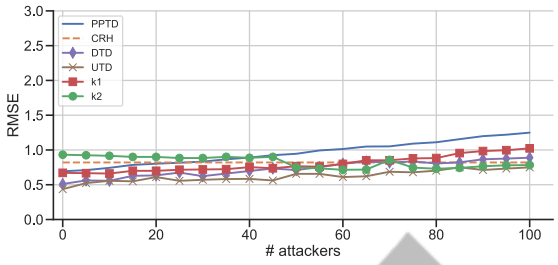
Analyzing the trends, we observe that as we increment the threshold, the RMSE values generally follow an upward trend. This rise gradually flattens out when the threshold reaches a high enough level. At this point, the system has already incorporated almost all data values into the computation, leaving out only the most extreme values. These extreme value are consistently rejected unless the threshold multiplier is ramped up to extraordinarily high levels.

In the following, for all comparisons with other work in the literature, we used the system with $k = 2$ and threshold $= 1$. We also present the results obtained by the system with $k = 1$, as baseline to discuss how the number of clusters affects the performance of BLIND against all proposed attacks.

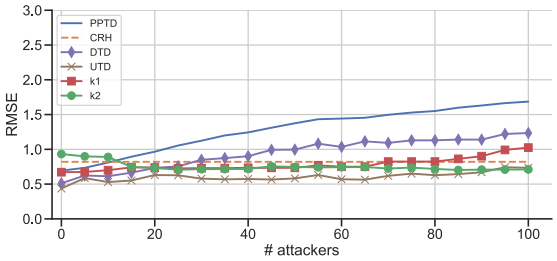
Figure 10 shows the result of the comparison. On the x-axis, the number of attackers varies from 0 to 100 (i.e., 50% of the total users), while the y-axis shows the RMSE values obtained by the compared systems. As in Figure 9, systems with lower values are the ones that perform best. In the graphical representation of our results, we have chosen



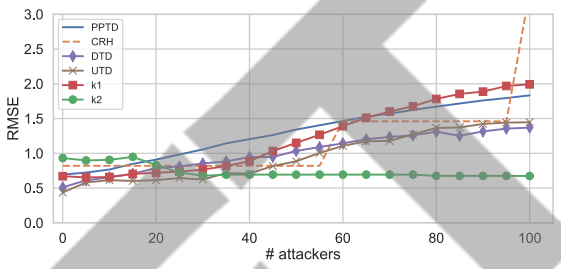
(a) Random attack.



(b) Slandering against one target.



(c) Slandering against two targets.



(d) Orchestraded slandering against one target.

Figure 10: RMSE obtained by the compared systems with *random*, *slandering* and *orchestrated slandering* attacks.

to omit error bars that would otherwise indicate the confidence intervals. This decision was driven by the fact that the confidence intervals associated with our experiments are so small that their inclusion would not contribute significantly to the visual interpretation of the results.

Figure 10a shows the behavior of the systems when subjected to the *random attack*. PPTD, representative of a wide range of truth discovery systems, performs the worst of all. As the number of attackers increases, its RMSE continues to grow steadily, meaning that its performance degrades in a scenario of increasing adversaries. In contrast, the outlier-resistant variant of CRH responds better to this attack, showing a surprising resilience to an increasing number of attackers up to a certain point, with a steady RMSE value. Unsurprisingly, the behavior of CRH is that of a step function, as shown in Figure 10a. This is consistent with the use of the weighted median instead of the mean as a criterion for selecting the final truth value. However, beyond 40 attackers, we can observe an uptick in RMSE, suggesting the limit of the system's resilience against adversaries.

The DTD system's RMSE increases steadily with the number of attackers, suggesting that it also struggles with increasing adversarial conditions, albeit at a more gradual rate than PPTD. The UTD system exhibits a surprising behavior. Although the RMSE shows an overall increasing trend with the number of attackers, it presents some fluctuations. Its performance appears more consistent compared to the others, but it also suffers from increasing adversaries. Our system in the $k2$ variant obtains the best results. In particular, as noted earlier, the system with $k = 1$ is slightly better than $k2$ when the number of attackers is 0 or otherwise very low. As the number of attackers increases, however, the system with $k = 2$ performs remarkably well, and succeeds in eliminating outliers effectively, greatly improving the quality of information over the other systems. Specifically, the $k1$ variant seems to maintain a rather steady progression, but beyond 45 attackers, it experiences a noticeable surge in RMSE values. Despite this, it performs better than PPTD and is somewhat comparable to DTD, under a high number of attackers.

The best performing system is undoubtedly the $k2$ variant of our system. It starts with a high RMSE at zero attackers, but rapidly improves as the number of attackers increases. It then maintains a steady and low RMSE for the rest of the attack range, making it the most resilient system to adversarial conditions. Its performance remains impressive, with slight fluctuations but no alarming surges, even in the face of a high number of attackers. This robustness, combined with a generally low RMSE, suggests that $k2$ is the superior performer under these conditions.

When varying the number of attackers, the mean RMSEs obtained are 1.649, 1.383, 1.104, 0.811, 1.201, and 0.693 for PPTD, CRH, DTD, UTD, $k1$, and $k2$, respectively. Thus, PPTD, CRH, DTD and UTD obtained, respectively, a mean RMSE that is 138%, 100%, 59%, and 17% higher than our system with $k = 2$.

Figures 10b and 10c show *slandering attacks* with one or two targets, respectively. Overall, the TD systems are less affected by this attack, compared to the random one. PPTD shows the highest vulnerability, with a high RMSE that escalates as the number of attackers increases. Comparing the two figures, moreover, we can notice that PPTD obtains better results when the slandering is against a single target (Figure 10b), while it presents a higher RMSE

Table 5: Comparison of the communication overhead of PPTD, EPTD and BLIND for each iteration.

Phase	PPTD	EPTD	BLIND
Weights update	$5nq + 3n + (h-1)(4q+2)$	$n(4q+3)$	$9nk - 3n$
Truth update	$nq + 2q(h-1) + 2(h-1)$	$nq + q$	$3nqk$

when the attack is against two targets (Figure 10c).

The CRH system seems to be highly resistant to slander attacks, taking advantage of the weighted median to ignore erroneous data even with many attackers, with its RMSE remaining consistent in the two scenarios. However, this value is relatively high compared to other systems even with zero attackers.

For both types of slander attacks, DTD's RMSE starts relatively low and increases as the number of attackers grows. This system seems to handle small number of attackers reasonably well but struggles as the number of attackers grows. UTD obtains the best performance against this type of attack. Its RMSE starts low and has a lower rate of increase compared to the other systems, indicating a higher resilience to slander attacks. As for k1 and k2, their results are better when the attack is against two targets, contrary to PPTD. In particular, the system with $k = 2$ performs better against the attack in Figure 10c when the number of attackers is low. Our baseline system, $k1$, displays good resilience to the increasing numbers of attackers. In the one-target slander attack, in particular, its RMSE increases slower than PPTD and DTD but faster than UTD. Focusing on our system of interest, $k2$'s RMSE starts relatively high compared to the other systems but displays great resilience against increasing attackers both in the one-target and two-target slander scenarios.

Finally, Figure 10d shows a more elaborate attack, namely *orchestrated slandering* against a single target. As described above, this attack involves malicious users colluding to lower the average rating of a particular book. Such scenarios, where several attackers work together towards a common goal, are often problematic for truth discovery systems for multiple reasons. First, in an orchestrated slandering attack, a large number of attackers collude to provide a consistent false rating. This means that TD systems cannot rely on the assumption that the truth is the opinion of the majority. Moreover, since our attackers are real users who have rated all books fairly except the target, their behavior may not significantly deviate from normal until the point of the attack. This can make it difficult for TD systems to identify these users as attackers.

The results clearly show that the RMSE increases in most systems as the number of attackers rises. This indicates that as more false information is introduced, it becomes increasingly difficult for these systems to discern the truth, leading to higher error rates.

Indeed, CRH is susceptible to this type of attack, as shown in the figure. Its classic step function behavior is also present here, although as the number of attackers increases, the RMSE of CRH increases dramatically. The abrupt increase in RMSE at 60 attackers implies a weakness in handling high volumes of coordinated false information. PPTD behaves similarly to the cases of unorchestrated slandering attacks, with its RMSE steadily increasing along with the number of attackers. Its inability to distinguish between true and false ratings in the context of orchestrated slander is highlighted here.

Table 6: Default values of the parameters used in the scalability experiments.

Parameter	Value
# of users	1000
# of clusters	3
# of tasks	10
Key size	1024 bit

The steady increase in RMSE for both DTD and UTD highlights the challenges they face in identifying and excluding the ratings from the orchestrated attack. They seem unable to discern outliers effectively when those outliers are part of a coordinated group.

Our baseline $k1$ system, while better than the previous ones, also exhibits a consistent rise in RMSE, demonstrating the big challenges posed by these orchestrated attacks and yielding results similar to those in Figure 10a (random attack). Notably, our $k2$ system outperforms all others, demonstrating its superior ability in maintaining high accuracy despite the increase in attackers. After an initial increase, it manages to reduce the RMSE significantly, suggesting an effective mechanism to cope with the high volume of false information. The fact that the RMSE remains mostly constant thereafter, even with increased attackers, demonstrates a robust resilience against this type of attack. This is due to its outlier detection and its approach that is different from the other systems.

On average, when varying the number of attackers, PPTD, CRH, DTD and UTD obtained, respectively, an RMSE that is 76%, 61%, 37%, and 29% higher than our system with $k = 2$. In our comprehensive investigation, we have conducted Analysis of Variance (one-way ANOVA) tests for all considered scenarios, involving the four distinct attack types. Each case presented a statistically significant difference among the performance of the truth discovery systems, as indicated by the low p-values obtained (typically much less than 0.05). This consistent statistical significance across all cases confirms the robustness of our experimental methodology and demonstrates that the differences observed in the RMSE between the truth discovery systems are not due to random chance, but are reflective of their performance characteristics.

7.2. Communication Overhead

In this section we will compare the communication overhead of BLIND with other relevant state-of-the-art approaches, namely PPTD (Miao et al., 2019) and EPTD (Xu et al., 2017), which are two of the systems that implement the truth discovery iterative algorithm of CRH. In this comparison, CRH, DTD and UTD are not included since they are not privacy-preserving systems.

Table 5 shows the communication overhead of the three systems during an iteration of their respective protocols. All values for PPTD and EPTD are taken from (Xu et al., 2017). Since BLIND implements a different truth discovery model than either PPTD or EPTD, we merged the first two rows from the original table to carry out the comparison. In BLIND, the "weights update" of PPTD and EPTD is comparable to the computation of user distances and the subsequent calculation of the argmin, for each user. The "truth update", on the other hand, is easily comparable to the centroids update phase of BLIND. All values listed are intended to be multiplied by the size of an encrypted value, which is not shown for the sake of presentation clarity. It is worth recalling that n denotes the number of users, q indicates the number of tasks, and k represents the number of clusters. PPTD uses an additional parameter h that represents the minimum number of users needed to decrypt values, as provided by the Threshold Pallier cryptosystem.

As written above, the "weights update" phase for BLIND includes both user distances and argmin calculations. The SP computes the encrypted version of user distances by itself, without involving the BH. No messages are exchanged in this phase, hence there is no communication overhead.

The argmin computation was extensively detailed in Section 5.3.2. For each user, it is necessary to compute the argmin of a vector of k elements. In terms of exchanged values, at the beginning of the protocol, the SP sends to the BH k blinded distances and k random values, for a total of $2k$ values. Next, the BH sends to the SP the two k-dimensional vectors $E_b^+(\alpha)$ and $E_s^+(\beta)$. The SP, at this point, performs $(k-1)$ secure comparisons, for a total of $3(k-1)$ exchanged values. Finally, the SP and the BH send a k-dimensional vector to each other, for a total of $2k$ values. Thus, the communication overhead of the argmin phase for a user is $6k + 3(k-1) = 9k - 3$ values. If we consider n users, the cumulative overhead is $9nk - 3n$ values exchanged. In the centroid update phase, qk multiplications are performed for each user to calculate Equation 22, while the summation is carried out locally by the SP. Each multiplication involves sending 3 values, as described in Section 4. When considering n users, then, the overall values exchanged are $3nqk$.

To make sure the comparision is fair, we need to estimate the magnitude of the parameters k and h, which are specific to BLIND and PPTD, respectively. The factor k represents the number of clusters, which is typically very small. Effectively, increasing the number of clusters does not always lead to better QoI, as shown by our experiments on the real world dataset in Section 7.1. In our analysis, the elbow method (Fig. 8) indicated 2 as the ideal number of clusters. In the following, we will therefore consider k as a constant. The parameter h of PPTD, on the other hand, can assume high values (proportional to the number of users n), since the security of the whole system increases the higher the value of h.

As regards the "weights update" phase, the asymptotic communication overhead of PPTD is $O(nq + hq)$, that of EPTD is $O(nq)$ and that of BLIND is $O(n)$. It is worth noticing that BLIND is the only system whose overhead is not affected by the number of tasks q, in this phase, making it the most efficient. Finally, in the "truth update" phase, the asymptotic communication overhead of PPTD is $O(nq+hq)$, while EPTD and BLIND both exhibit $O(nq)$ overhead.

7.3. Scalability of the System

In this section we will evaluate the performance of our system in terms of execution times. We will analyze the scalability of BLIND when varying some key parameters, i.e., the number of users, the number of clusters (k parameter of the K-Means), the number of tasks performed by users and the length of the encryption key in bit.

7.3.1. Methodology

In the following, we will compare several versions of our privacy preserving truth discovery system, which take advantage of different kinds of performance optimizations. We will examine five different system variants:

- *baseline*: basic system without optimizations;
- *fast_ dist*: system that efficiently calculates users' distances from each cluster, exploiting Equation 10. Compared to the *baseline* system, *fast_ dist* needs to ask users the encrypted version of their squared values, $E_b^+(v_{ij}^2)$, for each task, as described in Section 5.2, but this allows a significant reduction in execution times, as will be shown by the results;
- *fast_ argmin*: system that efficiently calculates the argmin (Protocol 4), instead of using the simpler but slower method mentioned in the same section;
- *fast_ outlier*: system that speeds up the outlier removal phase if the system has reached convergence. If this is the case, $fast\_outlier$ avoids recalculating the encrypted mininum indicator vector $E_b^+(A_i)$ for each user, since it will be unchanged from the last K-Means iteration; instead, $E_b^+(A_i)$ can be stored and reused from the previous step;
- *fully_ optimized*: system that exploits all the optimizations described above simultaneously, i.e., *fast_ dist*, *fast_ argmin* and *fast_ outlier*.

Table 7: Comparison of execution times of the different system phases under different optimizations with the default parameters reported in Table 6, i.e., 1000 users, 3 clusters, 10 tasks, and 1024 bit key size.

	baseline	fast_dist	fast_argmin	fast_outlier	f_optimized
K-Means	1.00x (332.3s)	0.61x	0.84x	1.00x	0.45x
Outlier removal	1.00x (114.6s)	0.48x	0.89x	0.22x	0.20x
Total time	1.00x (446.9s)	0.55x	0.86x	0.67x	0.34x

Synthetic data are generated according to a uniform distribution with values between 0 and 10. The default values of the parameters used in the experiments are given in Table 6. In each series of experiments we vary one of those parameters, leaving the others unchanged. In the results we report the average execution times of the various systems, normalized in such a way that the *baseline* system requires 1 computation unit when using the default parameters. For the 1.00x baseline system we also report the absolute execution times of our Python implementation on our test hardware. Note that by using higher-performance hardware or lower level languages such as C, the absolute times may decrease quite a bit. For example, our tests show that homomorphic operations on data performed in Python can take up to 6.81x the time required by an equivalent Paillier library in C.

7.3.2. Server-side Scalability Results

Table 7 compares all five systems with the default parameters that are reported in Table 6. In addition to the total execution time, we also present the partial time needed to calculate the K-Means and that for the removal of outliers, also normalized according to the *baseline* system. The K-Means phase performs a variable amount of iterations (generally between 2 and 4) until convergence, when no more changes from the previous iteration.

Unsurprisingly, each system is faster than the *baseline*, with different levels of performance. The *fast_ dist* optimization reduces the time taken for the K-Means phase by about 39%, making it faster than the baseline. The *fast_ argmin* optimization also speeds up this phase, but less effectively, reducing the time to 84% of the baseline. The *fast_ outlier* optimization makes no change to this phase compared to the baseline. This is because *fast_ outlier* only optimizes the outlier removal phase. The K-Means phase of *fast_ outlier* is therefore identical to the *baseline* system, as shown by its 1.0x execution time. The most significant improvement comes from the *fully_ optimized* system, which reduces the time taken to just 45% of the baseline. In the outlier removal phase, the *fast_ dist* optimization achieves a 52% reduction in time, while the *fast_ argmin* optimization only gives an 11% time reduction. The *fast_ outlier* and *fully_ optimized* optimizations perform particularly well in this phase, reducing the time to 22% and 20% of the baseline, respectively. Unsurprisingly, the *fully_ optimized* system that exploits all optimizations at once is the fastest one, with execution times of 0.45x, 0.20x and 0.34x for K-Means, outlier removal and total time, respectively. Thus, the *fully_ optimized* system is 3 times faster than the baseline. The outlier removal phase is the one that benefits the most from optimizations, improving the *baseline* execution time by a factor of 5.

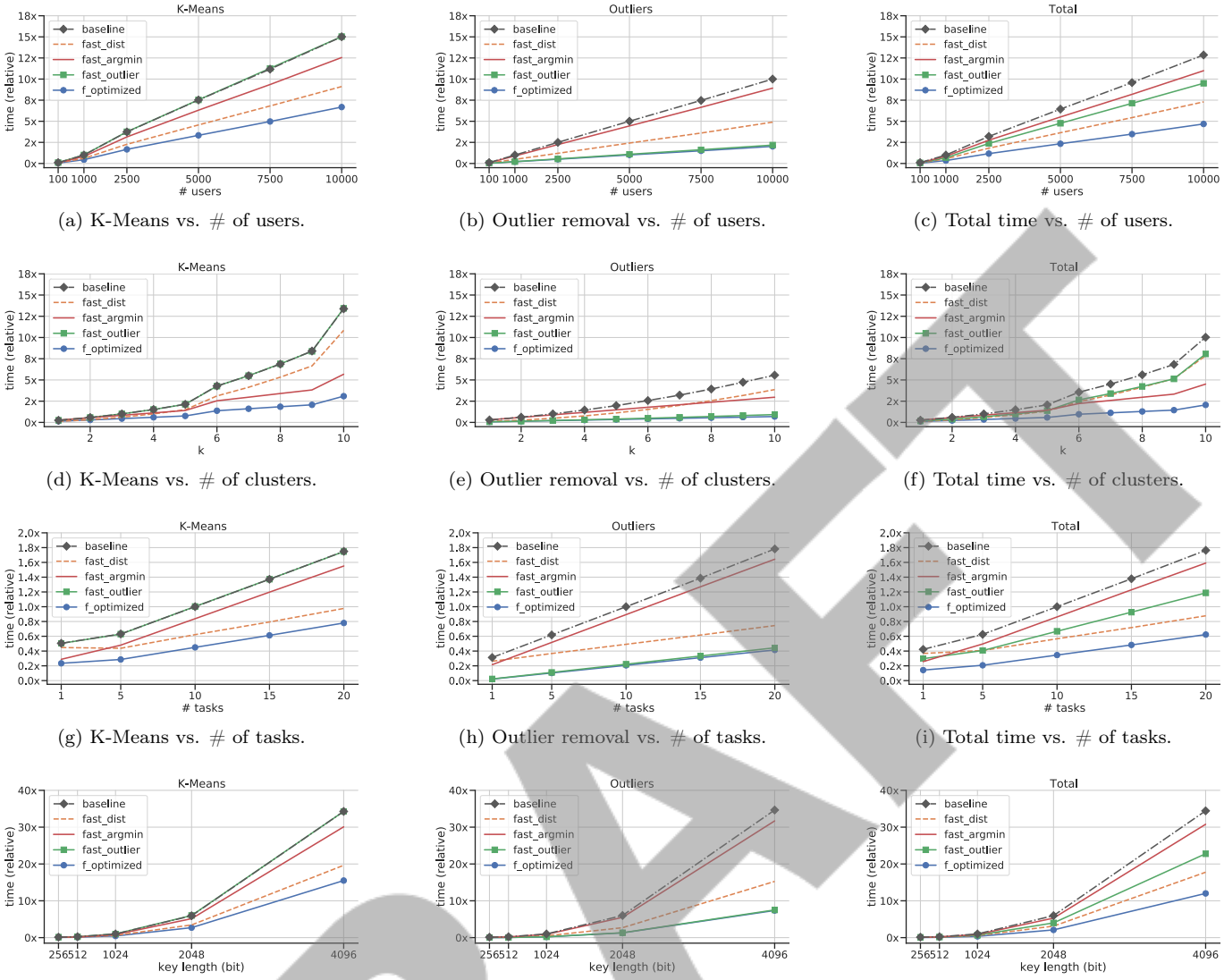
Figure 11 shows a comparison of the execution times of the five analyzed systems, by varying the selected parameters (i.e., number of users, number of clusters, number of tasks and key length) one at a time. Each group of graphs shows the average execution times for the K-Means and outlier removal phases, as well as the total time. As in the previous set of experiments, we have omitted error bars to avoid visual clutter, since the confidence intervals are low.

It is worth noticing that, in all cases, the *fully_ optimized* system is consistently the fastest, even when varying parameters. At the same time, as pointed out above, the *fast_ outlier* system is identical to *baseline* in the K-Means phase and close to *fully_ optimized* in the outlier removal phase.

Figures 11 (a-c) show the trend of execution times as the number of users increases from 100 to 10,000. We can observe that execution times increase linearly with respect to the number of users, for all systems. The *baseline* system consistently trails behind, suggesting that the optimizations in the other systems significantly improve execution times. Aside from the *fully_ optimized* system, as users increase, the single optimization with the greatest effect is *fast_ dist*. This is as expected, considering that the distance calculation times become more impactful as the number of users increases.

Surprisingly, while *fast_ argmin* is the least effective optimization as the number of users varies, it becomes very important as the number of clusters increases, as shown in Figures 11 (d-f). Once again, the *fully_ optimized* system consistently performs better than other systems across all phases and across all numbers of clusters, reinforcing the previous conclusion that the optimizations significantly improve system performance. Indeed, in this case, only the execution times of *fast_ argmin* and *fully_ optimized* follow a linear growth as the number of clusters increases, while the other systems follow a superlinear but subquadratic growth. This can be explained by the fact that the argmin calculation is $O(k)$, where k is the number of clusters, in the *fast_ argmin* and *fully_ optimized* systems, while it is $O(k^2)$ in other systems. Since the argmin calculation is only a portion of the total execution time, the result is correctly a superlinear but subquadratic growth.

The *fast_ dist* optimization performs comparatively bet-



(a) K-Means vs. # of users. (b) Outlier removal vs. # of users. (c) Total time vs. # of users.

(d) K-Means vs. # of clusters. (e) Outlier removal vs. # of clusters. (f) Total time vs. # of clusters.

(g) K-Means vs. # of tasks. (h) Outlier removal vs. # of tasks. (i) Total time vs. # of tasks.

(j) K-Means vs. key size. (k) Outlier removal vs. key size. (l) Total time vs. key size.

Figure 11: Experiments on the scalability of BLIND when varying different parameters.

ter when the number of clusters is low but shows a steeper increase in execution time as the number of clusters grows, especially noticeable in the K-Means phase. This suggests that this optimization might not scale as well with the increasing complexity of calculations related to more clusters. Both the *fast_ outlier* and *fast_ argmin* optimizations have less drastic increases in execution time with more clusters than *fast_ dist*, particularly noticeable in the Outlier removal phase.

The greater rise in the execution time of the K-Means phase from 9 to 10 clusters in Figure 11 (d) is explained by the increase in the number of iterations performed by the system to achieve convergence. In general, the number of iterations for convergence increases with the number of clusters, suggesting that the algorithms take longer to converge with more complex data structures (more clusters). Interestingly, this increase in iterations does not affect the *fully_ optimized* system as severely as the other systems, emphasizing its robustness. It can also be noted that the increase in the execution time of the outlier removal phase in Figure 11 (e) is more regular, since it is not influenced by the number of iterations performed.

Figures 11 (g-i) show the trend of execution times as the number of tasks performed by users varies from 1 to 20. The growth is linear for all systems, and *fast_ dist* is the most effective single optimization, after the *fully_ optimized* system. Surprisingly, the number of iterations performed by K-Means is less when the number of tasks is 5 than when it is 1, and this explains why execution times increase slightly or even decrease (system *fast_ dist* in Figure 11 (g)) in this particular case. This could imply that having more tasks can potentially help the algorithms to reach convergence faster, thus reducing the number of iterations. Once again, the outlier removal

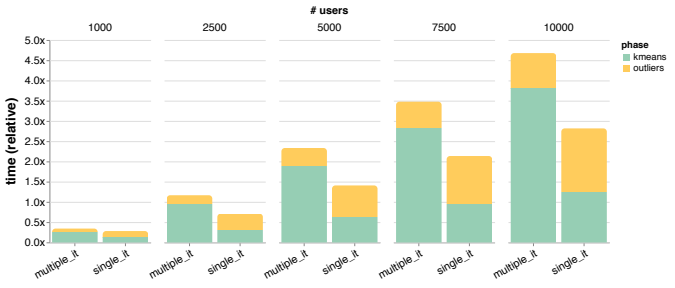


Figure 12: Relative importance, in terms of execution time, of the K-Means and outlier removal phases.

phase is not sensitive to the number of K-Means iterations and therefore shows a more regular growth.

Figures 11 (j-l) show that all systems display a marked increase in execution time as the key length increases across all phases (K-Means, outlier removal, and total). This indicates that encryption complexity heavily influences execution time. The *baseline* system execution time, for example, increases almost 35 times varying from a 1024 bit key to a 4096 bit one, while the *fully_optimized* system time is about 12x with the same key. Note that the execution time for the *baseline* system is about 35x in both the K-Means phase and the outlier removal phase, with a 4096 bit key. The *fully_optimized* system, on the other hand, shows an execution time of 15.5x in the K-Means phase and only 7.37x in the outlier removal phase. This is because the *fully_optimized* system performs much less cryptographic operations in the outlier removal phase with respect to other systems, and this has a positive effect on the total execution time, especially when such operations are extremely time-consuming, with a large cryptographic key. The 2048 bit key, on the other hand, results in an execution time of 5.94x for the *baseline* system and 2.1x for the *fully_optimized* system.

All other systems are in between these two extremes, as expected. In the K-Means phase, the *fast_outlier* has the most noticeable increases in execution time as key length grows (excluding the baseline), indicating that this system might be particularly sensitive to encryption complexity. In contrast, *fast_argmin* and *fast_dist* systems also increase but at a slightly lower rate, suggesting these optimizations handle increased encryption complexity better. In the outlier removal phase, the execution times for all systems rise steeply with key length, but *fully_optimized* remains the most efficient. *fast_dist* and *fast_argmin* have a slightly better response to key length increase than *fast_outlier* and *baseline*.

Finally, we analyze the relative importance, in terms of execution time, of the K-Means and outlier removal phases with respect to the complete truth discovery system. To this end, Figure 12 compares two versions of the *fully_optimized* system when varying the number of users from $1,000$ to $10,000$. The first system, called *multiple_it*, is the one that has been analyzed so far, where the K-Means performs multiple iterations until convergence is reached. The second system, called *single_it*, performs only a single iteration of the K-Means, regardless of convergence. The times reported are, once again, normalized, so that the total execution time of the *baseline* system with default parameters corresponds to 1 computation unit.

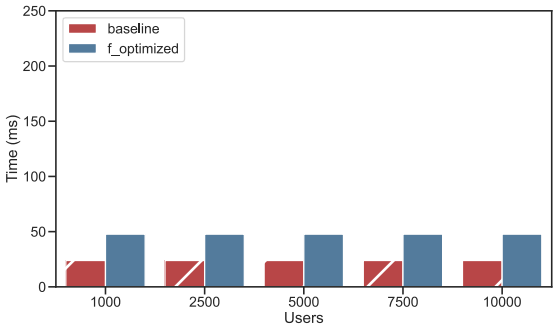
Consistent with the results obtained in the QoI optimization experiments, we observe that the K-Means phase generally takes longer than the outlier removal phase. Of course, this also depends on the number of iterations performed by the K-Means algorithm. As expected, the total execution time of the *single_it* system is always lower than the *multiple_it* one, since the number of iterations performed is different. At the same time, however, *multiple_it* is able to speed up the outlier removal phase, by applying the same optimizations of the *fast_outlier* system. Such optimizations are impossible if the K-Means has not reached convergence, as in the case of the *single_it* system. In this case, the outlier removal phase is as expensive as a normal K-Means iteration. This is highlighted by the *single_it* bars in Figure 12, where the outlier removal phase represents about half of the total time, regardless of the number of users. On the contrary, the *multiple_it* system bars show that its outlier removal phase has a much smaller impact on the total time. This is obvious, considering that *multiple_it* performs multiple iterations of the K-Means.

Indeed, for the *multiple_it* system, the K-Means phase consistently accounts for more than 70% of the total execution time, while the outlier removal phase contributes less than 30%. This shows the dominance of K-Means clustering in terms of computational complexity in the *multiple_it* system. On the contrary, in the *single_it* system, the execution time is distributed more evenly between the K-Means and outlier removal phases. The outlier removal phase takes up approximately 55% of the total time, on average, whereas the K-Means phase accounts for around 45%. This indicates that limiting the number of K-Means iterations significantly reduces its execution time, to the point where outlier detection becomes the most time-consuming operation.

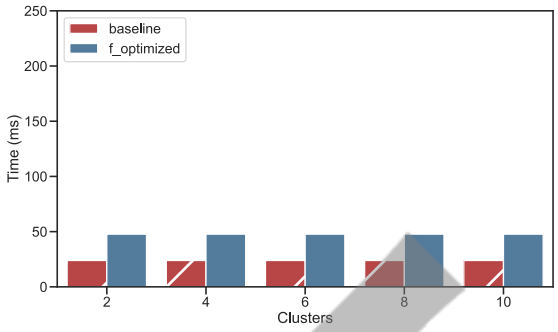
In conclusion, the optimizations performed by *multiple_it* are remarkable, since the execution times of the outlier removal phase are much lower in *multiple_it* with respect to *single_it*, when compared directly. This might be important if the SP wants to run the outlier removal system several times, for example by varying the threshold used to discard unreliable data, as shown in the QoI optimization experiments.

7.3.3. Client-side Scalability Results

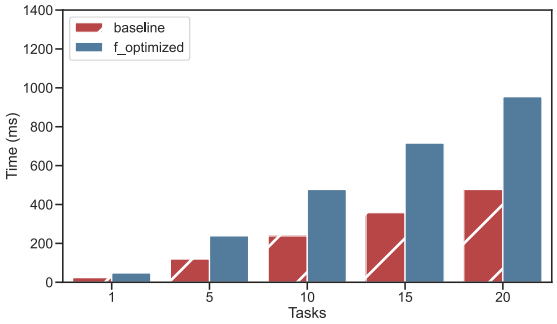
In this section we present another set of experiments we performed to test the scalability of the system from the perspective of end-users using their mobile devices. Experiments regarding client-side scalability are crucial for assessing the feasibility, adaptability, and future-proofing of a privacy-preserving truth discovery system used for mobile crowdsensing.



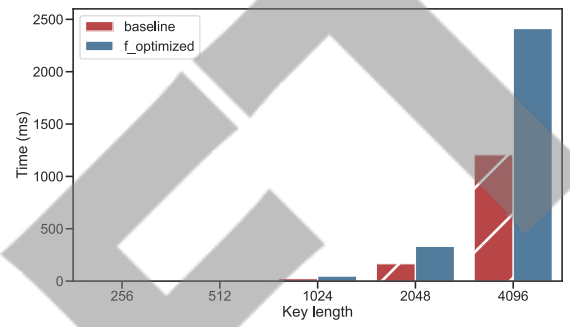
(a) Client-side scalability vs. number of users.



(b) Client-side scalability vs. number of clusters.



(c) Client-side scalability vs. number of tasks.



(d) Client-side scalability vs. key length.

Figure 13: Experiments on the scalability of BLIND on the client side.

We tested the execution time of encryption operations using Paillier's cryptosystem on a Samsung Galaxy S8, a 2017 smartphone. To perform the tests, we used an open-source benchmark suite available on GitHub[2], which is also compatible with iOS and Android.

In the following experiments, we compare two of the systems presented earlier, namely *baseline* and *fully_optimized*, since the other optimizations do not affect the execution time on the client side. Figures 13 (a-c) show the results of our experiments. Times are expressed in milliseconds.

We can immediately observe that, in Figures 13 (a) and (b), the execution time is about 24 ms and it is constant as the parameters (number of users and number of clusters), vary. At the same time, we can see that the *fully_optimized* system requires about 50 ms. This is as expected. Indeed, one of the strengths of BLIND is that users only have to encrypt their data once with their public key at the beginning of the protocol, with minimal effort. This makes the system very scalable on the client side, since many of the parameters that generally increase the client-side complexity of other privacy-preserving systems, such as the total number of users or the number of clusters, have no effect on the computations that each user has to carry out in BLIND.

From the user's point of view, the only thing that matters is the number of values to encrypt. In the *baseline* system, the number of values to encrypt is equal to the number of tasks, since the user only has to submit the encrypted version of its values, $E_b^+(v_{ij})$.

In order to compute encrypted distances from centroids more efficiently, as in the *fast_dist* and *fully_optimized* systems, it is necessary that the user also sends the encrypted version of his values squared, $E_b^+(v_{ij}^2)$, as described in Section 5. Since the number of values to encrypt is doubled, this generally results in doubling the time required on the client side, as well. However, since the number of values required from the user is generally small (on the order of a dozen), the overall time is still very low, and the system is still scalable and applicable to real-world scenarios.

This can be seen in Figure 13 (c), which shows a linear growth for both systems as the number of tasks increases. As argued above, 10 tasks is a reasonable number to focus on in a mobile crowdsensing application. With 10 tasks, the amount of time required by users is about 239 ms for the *baseline* system and 477 ms for the *fully_optimized* system. This is still quite reasonable and shows that the cryptographic operations required by BLIND are perfectly usable in a real crowdsensing application without affecting the user experience.

Finally, Figure 13 (d) shows the time required for different key lengths. As discussed above in the server-side scalability experiments, the length of the cryptographic keys used for the homomorphic operations has a significant impact on the execution times.

[2]https://github.com/snipsco/paillier-libraries-benchmarks

As the figure shows, the growth is exponential in this case. The encryption times with 256 and 512 bit keys are so small that the bars are not visible in the graph (i.e., about 2 ms and 7.4 ms for the *fully_ optimized* system, respectively). However, such key lengths do not provide adequate security. On the other hand, up to 2.4 seconds for encrypting a single value with a 4096-bit key is too much for a practical application (this would mean about 24 seconds for 10 tasks in a typical application).

As it stands, there is a trade-off between privacy and efficiency. Using long cryptographic keys provides better security but requires more computational resources. These client-side scalability experiments can help determine that 1024 bits is a good key length that balances privacy and efficiency, providing secure communication without burdening user devices.

With a 1024-bit key, the average time required for a typical crowdsensing application with 10 tasks would be less than 500 ms on the *fully_ optimized* system, which is quite reasonable considering that this is the only computational effort required from users and that newer mobile devices would further reduce this time.

8. Discussion

BLIND proposes an innovative solution to the challenges of privacy preserving truth discovery systems for mobile crowdsensing. Overall, this study answers multiple questions:

1. Is it possible to implement a system that improves the QoI of data collected through mobile crowdsensing while respecting the privacy of participants?
2. What is the communication overhead of such a system? Is the system scalable to real-world application scenarios?
3. Is the system scalable from a server-side perspective as various operational parameters change, such as number of users, number of tasks, cryptographic key length, etc.?
4. Is the system scalable from the perspective of the end-users as well, who are often using devices that have limited power and computing resources?

Answering these questions is critical to assessing the feasibility and usability of BLIND in real-world cases and to evaluate its possible integration with existing crowdsensing systems. Our extensive experimental evaluation covered such challenges, showing how BLIND solves them and how it compares with relevant work in the literature, by highlighting strengths and limitations of our approach.

The first issue that we analyzed was the choice of feasible parameters to tune our system. Although automatic and dynamic tuning of these parameters could be the subject of future research, our experiments showed that a threshold value hovering around 1 consistently delivers optimal performance across all system configurations (Figure 9). This suggests that at this threshold value, BLIND achieves a balance between including a sufficient amount of data for a significant analysis and excluding enough outliers to maintain the overall accuracy of the results. Thus, this particular threshold level could be considered a sweet spot for optimal system performance.

We then tried to answer question (1) by analyzing how BLIND can privately improve QoI in mobile crowdsensing systems. We compared BLIND (in two variants named $k1$ and $k2$) to both privacy-preserving and non-privacy-preserving TD systems in multiple scenarios with an increasing number of malicious users. By analyzing the results, we can draw interesting insights on their relative performances under varying adversarial conditions. Starting with the PPTD and CRH systems, we see a significant contrast in their resilience to an increase in the number of attackers. This is both surprising and interesting, since PPTD is inspired by CRH, and it means that the outlier-resistant version of CRH proposed by its authors is remarkably effective in improving the system.

The performance patterns of both the UTD and $k2$ systems, in terms of RMSE, offer an interesting comparison. Initially, the UTD system starts with the lowest RMSE among all systems, indicating the highest accuracy in truth discovery. However, as the number of attackers increases, its RMSE also rises, suggesting that the system's performance decreases in the face of adversarial conditions.

Contrastingly, the $k2$ system initially starts with a higher RMSE, particularly when there are zero attackers. This could be due to the system's outlier removal method, which might initially remove useful data points rather than actual outliers in the absence of attackers.

However, as the number of attackers increases, an interesting shift occurs. The $k2$ system improves and its RMSE decreases, settling to a consistently low value that it maintains despite further increase in attackers. The outlier removal method, which initially seemed a liability, becomes an asset when dealing with adversarial conditions.

UTD performed slightly better than all other systems, including BLIND, for a specific attack type, namely slandering against one or two targets. A common challenge for both attacks is the randomness and independence in the choice of targets by the attackers. This makes it difficult for the systems to discern patterns and effectively counter the attacks.

When comparing the performance of the $k2$ and UTD truth discovery systems under slandering attacks, it is important to note that, while UTD may show slightly superior performance in terms of RMSE, the performance of $k2$ remains quite impressive. Indeed, despite the slander attacks, $k2$ maintains a relatively stable RMSE even as the number of attackers increases. Moreover, it is crucial to highlight the different aims of the two systems. While UTD focuses solely on truth discovery, k2 serves a dual purpose by also aiming to preserve the privacy of its users. The necessity to balance these two often conflict-

ing objectives inevitably impacts the ability of k2 to match the RMSE values achieved by UTD. In an era where privacy concerns are becoming increasingly important, while systems such as UTD might provide slightly superior accuracy, they do so at the cost of privacy. By managing to achieve remarkable accuracy while preserving privacy, the $k2$ system offers a more balanced solution for private truth discovery.

To answer question (2), we evaluated the communication overhead of BLIND, comparing it with other relevant state-of-the-art approaches. The novel approach adopted by BLIND allowed it to outperform the competition in this comparison, exhibiting low overhead both for updating the required weights and updating the truth values, with asymptotic communication overhead of $O(n)$ and $O(nq)$, respectively.

To answer question (3), we performed a series of tests regarding the server-side scalability of BLIND, comparing different versions of the system with various optimizations. The *fully_optimized* version of the system clearly stands out in terms of overall efficiency, providing significant time reductions in both the K-Means and outlier removal phases. While the other optimizations also provide improvements in speed, their effects are not as consistent or as effective. In general, the linear increase of execution time with the rise in user count indicates the scalability of these systems. However, this also underlines the need for performance optimization, particularly when dealing with large user bases.

The experiments on key length showed a possible limitation of the system, since the increase in execution time is exponential in this case. As mentioned above, it is certainly necessary to find an appropriate compromise between security and efficiency in this case. A 1024-bit key seems to be the best choice, providing both satisfactory security and execution times compatible with real applications. However, it is worth noting that the computations performed on the server side can be accelerated by scaling the servers both horizontally and vertically, and that we are talking about batch computations which have no real-time requirements. Thus, we can conclude that the runtimes achieved by BLIND are perfectly acceptable in real application scenarios.

As a result of BLIND's optimization efforts, the system *fully_optimized* is on average 3 times faster than the baseline. Nevertheless, future research efforts could be directed at finding new optimizations to further improve the efficiency of BLIND.

These efforts will also focus on client-side performance optimization, which answers the last of our questions about the feasibility of BLIND for users with mobile devices. Indeed, any significant computational load on these devices, such as running complex algorithms, can drastically affect battery life and slow down the device, leading to a poor user experience. As a result, users may become disinclined to use the service, which in turn, can affect the amount of data being collected. In particular, client-side scalability ensures that users with older devices can still participate. This broadens the potential user base, making the system more accessible, which is particularly important for crowdsensing applications.

The results are very encouraging. BLIND requires minimal computational effort from end-users, who only need to encrypt their data at the beginning of the protocol and do not need to stay connected or perform additional operations. The system proved to be extremely scalable from the client side. Again, a potential limitation of the system is the sharp rise in complexity due to increasing the cryptographic key size, but 1024-bit keys have proven to be perfectly adequate to maintain execution times in the range of a few hundred milliseconds for a typical crowdsensing application.

9. Conclusions

In this work we studied the possibility of improving QoI of the observations provided by users, without knowing either the content of the data or the users who actually provided it. To accomplish this task, we proposed BLIND, an open source privacy-preserving truth discovery system for mobile crowdsensing.

Encrypted data sent by users is anonymously grouped in clusters so as to preserve privacy through a modified version of the K-Means algorithm that exploits homomorphic encryption techniques. The result obtained is used to discard outliers which are not close enough to any of the cluster centroids. Finally, a single truth value for each activity is calculated as a function of the centroids.

We demonstrated that BLIND is secure and that no information about users is disclosed to any of the parties involved in the protocol. We released the source code of BLIND and made it publicly available.

We also extensively evaluated the performances of BLIND, comparing it with other state-of-the-art work by using a real world dataset. Results showed that BLIND outperforms other systems in reducing the impact of four security attacks. We further investigated the communication overhead of BLIND and compared it with other relevant work. Finally, we evaluated the scalability of our approach both server-side and client-side by using a synthetic dataset to test the system under challenging conditions.

Our solution offers interesting perspectives for the development of future truth discovery systems and is a significant step forward in safeguarding participants' privacy.

Incorporating the concept of user reputation into the model, while preserving their privacy through a reputation management system, may be an interesting area for future research, as it would further enhance the QoI of the truth discovery system. Other possible directions for future work include allowing BLIND to dynamically reconfigure itself according to the context, and trying to further optimize the system, both server-side and client-side, to make it even more scalable.

Acknowledgments

This research is partially funded by the European Union - PON Ricerca e Innovazione 2014-2020 - DM 1062/2021 and partially funded by the NSF grants in the US through awards SaTC-2030624, CNS-1818942, and DGE-1433659.

References

Agate, V., De Paola, A., Ferraro, P., Lo Re, G., Morana, M., 2021a. Secureballot: A secure open source e-voting system. Journal of Network and Computer Applications 191, 103165. doi:https://doi.org/10.1016/j.jnca.2021.103165.

Agate, V., De Paola, A., Lo Re, G., Morana, M., 2021b. A simulation software for the evaluation of vulnerabilities in reputation management systems. ACM Transactions on Computer Systems 37. doi:10.1145/3458510.

Akyildiz, I.F., Kak, A., Nie, S., 2020. 6G and beyond: The future of wireless communications systems. IEEE Access 8, 133995–134030. doi:10.1109/ACCESS.2020.3010896.

Al-Janabi, S., Al-Barmani, Z., 2023. Intelligent multi-level analytics of soft computing approach to predict water quality index (im12cp-wqi). Soft Computing 27, 7831–7861.

Al-Janabi, S., Al-Janabi, Z., 2023. Development of deep learning method for predicting dc power based on renewable solar energy and multi-parameters function. Neural Comput. Appl. 35, 15273–15294. doi:10.1007/s00521-023-08480-6.

Al-Janabi, S., Alkaim, A., 2022. A novel optimization algorithm (lion-ayad) to find optimal dna protein synthesis. Egyptian Informatics Journal 23, 271–290. doi:https://doi.org/10.1016/j.eij.2022.01.004.

Al-Janabi, S., Alkaim, A., Al-Janabi, E., Aljeboree, A., Mustafa, M., 2021. Intelligent forecaster of concentrations (pm2.5, pm10, no2, co, o3, so2) caused air pollution (ifcsap). Neural Comput. Appl. 33, 14199–14229. doi:10.1007/s00521-021-06067-7.

Al-Janabi, S., Alkaim, A.F., 2020. A nifty collaborative analysis to predicting a novel tool (drflls) for missing values estimation. Soft Comput. 24, 555–569. doi:10.1007/s00500-019-03972-x.

Al-Janabi, S., Alkaim, A.F., Adel, Z., 2020a. An innovative synthesis of deep learning techniques (dcapsnet & dcom) for generation electrical renewable energy from wind energy. Soft Computing 24, 10943–10962.

Al-Janabi, S., Mohammad, M., Al-Sultan, A., 2020b. A new method for prediction of air pollution based on intelligent computation. Soft Comput. 24, 661–680. doi:10.1007/s00500-019-04495-1.

Al-Janabi, S., S. Mohammed, G., Abbas, T., 2023. Main challenges (generation and returned energy) in a deep intelligent analysis technique for renewable energy applications. Iraqi Journal For Computer Science and Mathematics 4, 34–47. doi:10.52866/ijcsm.2023.02.03.004.

Al-Janabi, Z.K., Al-Janabi, S., 2022. An efficient predictor of renewable energy based on deep learning technique (dgbm) and multi-objectives optimization function, in: 2022 Iraqi International Conference on Communication and Information Technologies (IICCIT), pp. 44–50. doi:10.1109/IICCIT55816.2022.10010380.

Alkaim, A.F., Al_Janabi, S., 2020. Multi objectives optimization to gas flaring reduction from oil production, in: Farhaoui, Y. (Ed.), Big Data and Networks Technologies, Springer International Publishing, Cham. pp. 117–139.

Barnwal, R.P., Ghosh, N., Ghosh, S.K., Das, S.K., 2019. Publish or drop traffic event alerts? quality-aware decision making in participatory sensing-based vehicular cps. ACM Transactions on Cyber-Physical Systems 4, 1–28.

Becnel, T., Tingey, K., Whitaker, J., Sayahi, T., Lê, K., Goffin, P., Butterfield, A., Kelly, K., Gaillardon, P.E., 2019. A distributed low-cost pollution monitoring platform. IEEE Internet of Things Journal 6, 10738–10748. doi:10.1109/JIOT.2019.2941374.

Bettini, C., Riboni, D., 2015. Privacy protection in pervasive systems: State of the art and technical challenges. Pervasive and Mobile Computing 17, 159–174.

Bhattacharjee, S., Ghosh, N., Shah, V.K., Das, S.K., 2018. Qnq: Quality and quantity based unified approach for secure and trustworthy mobile crowdsensing. IEEE Transactions on Mobile Computing 19, 200–216.

Cecaj, A., Mamei, M., Zambonelli, F., 2016. Re-identification and information fusion between anonymized cdr and social network data. Journal of Ambient Intelligence and Humanized Computing 7.

Cheng, Y., Ma, J., Liu, Z., Wu, Y., Wei, K., Dong, C., 2022. A lightweight privacy preservation scheme with efficient reputation management for mobile crowdsensing in vehicular networks. IEEE Transactions on Dependable and Secure Computing .

Chorppath, A.K., Alpcan, T., 2013. Trading privacy with incentives in mobile commerce: A game theoretic approach. Pervasive and Mobile Computing 9, 598–612.

Fontaine, C., Galand, F., 2007. A survey of homomorphic encryption for nonspecialists. EURASIP Journal on Information Security 2007.

Goldreich, O., Micali, S., Wigderson, A., 1987. How to play any mental game, in: Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC, ACM. pp. 218–229.

Jäschke, A., Armknecht, F., 2018. Unsupervised machine learning on encrypted data, in: International Conference on Selected Areas in Cryptography, Springer. pp. 453–478.

Kadhuim, Z.A., Al-Janabi, S., 2023. Intelligent deep analysis of dna sequences based on ffgm to enhancement the performance and reduce the computation. Egyptian Informatics Journal 24, 173–190. doi:https://doi.org/10.1016/j.eij.2023.02.004.

Kang, Y., Liu, A., Xiong, N.N., Zhang, S., Wang, T., Dong, M., 2023. Dtd: An intelligent data and bid dual truth discovery scheme for mcs in iiot. IEEE Internet of Things Journal , 1–1doi:10.1109/JIOT.2023.3292920.

Khan, F., Rehman, A.U., Zheng, J., Jan, M.A., Alam, M., 2019. Mobile crowdsensing: A survey on privacy-preservation, task management, assignment models, and incentives mechanisms. Future Generation Computer Systems 100, 456–472.

Khan, W.Z., Xiang, Y., Aalsalem, M.Y., Arshad, Q., 2013. Mobile phone sensing systems: A survey. IEEE Communications Surveys Tutorials 15, 402–427. doi:10.1109/SURV.2012.031412.00077.

Kim, J.W., Edemacu, K., Jang, B., 2022. Privacy-preserving mechanisms for location privacy in mobile crowdsensing: A survey. Journal of Network and Computer Applications 200, 103315. doi:https://doi.org/10.1016/j.jnca.2021.103315.

Lagendijk, R.L., Erkin, Z., Barni, M., 2012. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. IEEE Signal Processing Magazine 30.

Li, Q., Cao, G., La Porta, T.F., 2013. Efficient and privacy-aware data aggregation in mobile sensing. IEEE Transactions on Dependable and Secure Computing 11, 115–129.

Li, Y., Li, Q., Gao, J., Su, L., Zhao, B., Fan, W., Han, J., 2016. Conflicts to harmony: A framework for resolving conflicts in heterogeneous data by truth discovery. IEEE Transactions on Knowledge and Data Engineering 28, 1986–1999. doi:10.1109/TKDE.2016.2559481.

Liao, D., Li, H., Sun, G., Zhang, M., Chang, V., 2018. Location and trajectory privacy preservation in 5g-enabled vehicle social network services. Journal of Network and Computer Applications 110, 108–118. doi:https://doi.org/10.1016/j.jnca.2018.02.002.

Liu, X., Li, S., Chen, X., Xu, G., Zhang, X., Zhou, Y., 2017. Efficient solutions to two-party and multiparty millionaires' problem. Security and Communication Networks 2017.

Liu, Y., Kong, L., Chen, G., 2019. Data-oriented mobile crowdsensing: A comprehensive survey. IEEE communications surveys & tutorials 21, 2849–2885.

Luo, T., Kanhere, S.S., Huang, J., Das, S.K., Wu, F., 2017. Sustainable incentives for mobile crowdsensing: Auctions, lotteries, and trust and reputation systems. IEEE Communications Magazine 55, 68–74.

Miao, C., Jiang, W., Su, L., Li, Y., Guo, S., Qin, Z., Xiao, H.,

Gao, J., Ren, K., 2019. Privacy-preserving truth discovery in crowd sensing systems. ACM Transactions on Sensor Networks 15. doi:10.1145/3277505.

Miao, C., Su, L., Jiang, W., Li, Y., Tian, M., 2017. A lightweight privacy-preserving truth discovery framework for mobile crowd sensing systems, in: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, pp. 1–9. doi:10.1109/INFOCOM.2017.8057114.

Mohammed, G.S., Al-Janabi, S., 2022. An innovative synthesis of optmization techniques (fdire-gsk) for generation electrical renewable energy from natural resources. Results in Engineering 16, 100637. doi:https://doi.org/10.1016/j.rineng.2022.100637.

Mohassel, P., Rosulek, M., Trieu, N., 2019. Practical privacy-preserving k-means clustering. Cryptology ePrint Archive, Paper 2019/1158. URL: https://eprint.iacr.org/2019/1158. https://eprint.iacr.org/2019/1158.

Neverova, N., Wolf, C., Lacey, G., Fridman, L., Chandra, D., Barbello, B., Taylor, G., 2016. Learning human identity from motion patterns. IEEE Access 4, 1810–1820. doi:10.1109/ACCESS.2016.2557846.

Paillier, P., 1999. Public-key cryptosystems based on composite degree residuosity classes, in: International conference on the theory and applications of cryptographic techniques, Springer.

Peeters, R., 2003. The maximum edge biclique problem is np-complete. Discrete Applied Mathematics 131, 651–654. doi:https://doi.org/10.1016/S0166-218X(03)00333-0.

Rane, S., Boufounos, P.T., 2013. Privacy-preserving nearest neighbor methods: Comparing signals without revealing them. IEEE Signal Processing Magazine 30.

Restuccia, F., Ferraro, P., Sanders, T.S., Silvestri, S., Das, S.K., Lo Re, G., 2018a. FIRST: A framework for optimizing information quality in mobile crowdsensing systems. ACM Transactions on Sensor Networks (TOSN) 15, 1–35.

Restuccia, F., Ferraro, P., Silvestri, S., Das, S.K., Lo Re, G., 2018b. IncentMe: effective mechanism design to stimulate crowdsensing participants with uncertain mobility. IEEE Transactions on Mobile Computing 18, 1571–1584.

Santani, D., Do, T.M.T., Labhart, F., Landolt, S., Kuntsche, E., Gatica-Perez, D., 2018. Drinksense: Characterizing youth drinking behavior using smartphones. IEEE Transactions on Mobile Computing 17, 2279–2292. doi:10.1109/TMC.2018.2797901.

Shehada, D., Yeun, C.Y., Jamal Zemerly, M., Al-Qutayri, M., Al-Hammadi, Y., Hu, J., 2018. A new adaptive trust and reputation model for mobile agent systems. Journal of Network and Computer Applications 124, 33–43. doi:https://doi.org/10.1016/j.jnca.2018.09.011.

Shen, Y., Yang, W., Li, L., Huang, L., 2017. Achieving fully privacy-preserving private range queries over outsourced cloud data. Pervasive and Mobile Computing 39, 36–51.

Sun, J., Li, J., Gao, H., Wang, H., 2018. Truth discovery on inconsistent relational data. Tsinghua Science and Technology 23, 288–302. doi:10.26599/TST.2018.9010004.

Wan, M., McAuley, J.J., 2018. Item recommendation on monotonic behavior chains, in: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys, ACM. pp. 86–94. doi:10.1145/3240323.3240369.

Xiao, H., Wang, S., 2023. A joint maximum likelihood estimation framework for truth discovery: A unified perspective. IEEE Transactions on Knowledge and Data Engineering 35, 5521–5533. doi:10.1109/TKDE.2022.3173911.

Xu, G., Li, H., Tan, C., Liu, D., Dai, Y., Yang, K., 2017. Achieving efficient and privacy-preserving truth discovery in crowd sensing systems. Computers & Security 69.

Yao, A.C., 1982. Protocols for secure computations, in: 23rd annual symposium on foundations of computer science (sfcs 1982), IEEE. pp. 160–164.

Yu, R., Oguti, A.M., Ochora, D.R., Li, S., 2022. Towards a privacy-preserving smart contract-based data aggregation and quality-driven incentive mechanism for mobile crowdsensing. Journal of Network and Computer Applications 207, 103483. doi:https://doi.org/10.1016/j.jnca.2022.103483.

Yuan, J., Tian, Y., 2017. Practical privacy-preserving mapreduce based k-means clustering over large-scale dataset. IEEE transactions on cloud computing 7, 568–579.

Zhang, C., Xu, C., Zhu, L., Li, Y., Zhang, C., Wu, H., 2020a. An efficient and privacy-preserving truth discovery scheme in crowdsensing applications. Computers & Security 97, 101848. doi:https://doi.org/10.1016/j.cose.2020.101848.

Zhang, C., Zhu, L., Xu, C., Liu, X., Sharif, K., 2019a. Reliable and privacy-preserving truth discovery for mobile crowdsensing systems. IEEE Transactions on Dependable and Secure Computing , 1–1doi:10.1109/TDSC.2019.2919517.

Zhang, C., Zhu, L., Xu, C., Ni, J., Huang, C., Shen, X.S., 2020b. Efficient and privacy-preserving non-interactive truth discovery for mobile crowdsensing, in: GLOBECOM 2020 - 2020 IEEE Global Communications Conference, pp. 1–6. doi:10.1109/GLOBECOM42002.2020.9322483.

Zhang, C., Zhu, L., Xu, C., Sharif, K., Du, X., Guizani, M., 2019b. Lptd: Achieving lightweight and privacy-preserving truth discovery in ciot. Future Generation Computer Systems 90, 175–184. doi:https://doi.org/10.1016/j.future.2018.07.064.

Zhang, C., Zhu, L., Xu, C., Sharif, K., Liu, X., 2019c. Pptds: A privacy-preserving truth discovery scheme in crowd sensing systems. Information Sciences 484, 183–196. doi:https://doi.org/10.1016/j.ins.2019.01.068.

Zhao, C., Yang, S., McCann, J.A., 2021. On the data quality in privacy-preserving mobile crowdsensing systems with untruthful reporting. IEEE Transactions on Mobile Computing 20, 647–661. doi:10.1109/TMC.2019.2943468.

Zheng, Y., Duan, H., Wang, C., 2018. Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing. IEEE Transactions on Information Forensics and Security 13.

Zheng, Y., Duan, H., Yuan, X., Wang, C., 2017. Privacy-aware and efficient mobile crowdsensing with truth discovery. IEEE Transactions on Dependable and Secure Computing .

Zigomitros, A., Casino, F., Solanas, A., Patsakis, C., 2020. A survey on privacy properties for data publishing of relational data. IEEE Access 8, 51071–51099. doi:10.1109/ACCESS.2020.2980235.