SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2022, Vol. 65 152106:1–152106:14
https://doi.org/10.1007/s11432-020-2940-8

Labrador: towards fair and auditable data sharing in cloud computing with long-term privacy

Xiaojie GUO[1], Jin LI[2*], Zheli LIU[1*], Yu WEI[1], Xiao ZHANG[3] & Changyu DONG[4]

[1]*College of Cyber Science, Nankai University, Tianjin* 300350*, China;*
[2]*Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangzhou* 510006*, China;*
[3]*School of Mathematics and Systems Science, Beihang University, Beijing* 100191*, China;*
[4]*School of Computing, Newcastle University, Newcastle upon Tyne* NE1 7RU*, UK*

Received 1 March 2020/Revised 8 May 2020/Accepted 4 June 2020/Published online 16 December 2021

Abstract Data are one of the most important sources of power that drives the world today. However, aggregating data is not an easy task with increasing legal regulations and concerns from users about their data privacy, and therefore incentives might be needed to encourage data sharing. In this paper, we present Labrador (LB), a system to handle the above problems. Our result demonstrates long-term privacy that reveals only an analytic result to the data analyst. An analytic task is delegated to clouds, which holds users' homomorphically encrypted data. We develop a lightweight verifiable blind decryption technique over the linearly homomorphic encryption scheme to verify the final result. Thus, its verifiability and blindness rely on over-determined and under-determined systems, respectively. To support incentives in data sharing, we leverage smart contract to realize binding contracts between mutually distrusted parties. In the game theory model with a non-collusion assumption, Labrador is secure against any rational adversary. Our evaluation demonstrates that the computational overhead for the data analyst and the data owner is insignificant (i.e., only a few seconds and milliseconds, respectively).

Keywords data sharing, homomorphic encryption, blockchain, smart contract, game theory

Citation Guo X J, Li J, Liu Z L, et al. Labrador: towards fair and auditable data sharing in cloud computing with long-term privacy. Sci China Inf Sci, 2022, 65(5): 152106, https://doi.org/10.1007/s11432-020-2940-8

1 Introduction

Cloud computing is a significant platform for data analytics owing to its huge computing and storage resources. With the advent of cloud computing, data analytics gets deployed on a large scale. Thus, it is one of the most significant innovations pushing society forward nowadays. However, data analytics often require a large amount of high-quality training data, such as shared records of patients for disease research and shopping history for advertisement.

Given users' ever-increasing concern about data privacy and existing legal regulations (e.g., general data protection regulation (GDPR)), data analytics, let alone data sharing in the cloud computing setting be challenging. The recent approaches to achieve data aggregation; whereas, preserving privacy is to ensure data owner grants data analysts the right to perform various analytic tasks under specified legal terms. Though, this practice is often not sufficient in various ways. First, it is only the data analyst who performs data aggregation could analyze these data, which implies that the data analyst may not upload these data to the clouds controlled by a third party. For resource-constrained data analysts, therefore, it is almost impossible to acquire the analytic result. Second, it is fair that the data owner should be rewarded by the data analyst for sharing data. Nevertheless, since the data owner does not participate in the analytic process and cannot audit the usage of her data, the data analyst may refuse to pay the reward. Finally, the privacy of the data owner is not always guaranteed since the data owner hands over her control of the data to the analyst, whose behavior is uncertain.

* Corresponding author (email: jinli71@gmail.com, liuzheli@nankai.edu.cn)

In some cases, a data owner might not wish to share her data permanently because of the privacy problems. For example, the medical records of a patient ought to be available before its expiry date only. In such situations, the patient should have the right to stop his/her data sharing immediately if required to do so. To address this problem, traditional paradigms [1–6] adopt access control techniques (e.g., attribute-based encryption) to dynamically revoke undesired parties. The revoked parties cannot decrypt the shared data anymore, which preserves the privacy of shared data. Yet, each of these techniques has its drawbacks. Then, making a local copy of the dataset is a common practice for many users, which greatly improves performance if data access latency is crucial for local machine learning tasks. As a result, even if the access right is revoked, a malicious adversary can still have access to the shared data or its copy. Moreover, most of the existing systems [1–6] do not consider the possibility that data republished by third parties could be potentially misused.

We refer to this feature of the conventional data sharing paradigm as short-term privacy. The term "short-term" indicates that the privacy of the data owner is guaranteed only if the plaintext data are revealed to an honest third party. However, when the party republishes the data or gets compromised, data privacy can no longer be guaranteed.

In this paper, we introduce the concept of long-term privacy to overcome the stated drawbacks using traditional paradigms. Notice, the delegated parties use the shared data for data analytics, so the long-term privacy ought to guarantee that the knowledge learned by delegated parties in data sharing is no more than the analytic result. We argue that this condition is stricter comparing to the current one since the latter allows for access to shared plaintext data by decryption (i.e., full knowledge). Long-term privacy is more applicable in practice given that no one is willing to hand over full control of their sensitive data to others.

Challenges. The primary challenge in this paper is to achieve long-term privacy in data sharing, which distinguishes our work from current research. The solution is based on homomorphic encryption (HE) [7–13], a popular technique for secure computation [14–17]. Specifically, shared data can be encrypted with a HE key controlled by the data owner. Then, these data can be outsourced to powerful cloud servers where the analytic task specified by the data analyst can be performed. Finally, the data owner decrypts the analytic result and forwards it to the data analyst. Notice that long-term privacy is fulfilled given that (i) no dataset-level leakage occurs in the data processing and (ii) the analytic result in plaintext is known only to the data analyst.

Unfortunately, to make the aforementioned HE-based solution secure and fair enough, we have to address a few subsequent challenges.

• Analytic tasks performed by the cloud should be verified. Since the HE evaluation is resource consuming, we outsourced this task to cloud servers. An immediate concern is how to ensure the delegated cloud server has correctly performed the analytic task and, thus, that the derived result is not crafted.

• Decryption by the data owner should be verified. Even though the cloud evaluates the analytic task correctly, dishonest decryption performed by the data owner may still lead to meaningless results.

• Data owner should not know the analytic result. Notice that the data owner did not pay to obtain this analytic result. For fairness, the owner should never for any reason take advantage of his/her decryption privilege to obtain this result for free.

Contributions. In this paper, we address the three subsequent challenges listed above and thus, realize a data-sharing scheme with guaranteed long-term privacy. In particular, from a technical perspective, we develop the following methods to overcome these challenges.

• Verifiable HE evaluation by clouds. We develop a mechanism, which applies replication-based insight to outsource data analysis, to mitigate the dishonest evaluation of clouds. That is, the data owner possessing the decryption key is involved in comparing the two analytic results returned by two clouds, accordingly. The two results are supposed to be identical if they are computed correctly. The data owner is encouraged to expose the dishonest behavior of clouds and will be rewarded for its well-grounded "impeachment". This contribution addresses the first challenge.

• Verifiable and blinded HE decryption by the data owner. We design a novel and lightweight verifiable blind decryption (VBD) technique over the HE. This technique ensures that the data analyst can verify whether the data owner decrypts the result correctly; whereas, the result remains unknown to the data owner. The VBD technique involves a system of equations, making it over-determined to the data analyst but under-determined to the data owner. This contribution enables the data analyst to verify the HE decryption launched by the data owner and leaves the analytic result unknown to the data owner. Our VBD technique tackles the remaining two challenges.

Table 1 Comparison of features supported by some state-of-the-art paradigms

Schemes	Decryption privilege	Resist potential dataset leakage?	Free of ciphertext/key updating?	Incentive mechanism	Auditability
Xu et al. [1]	Authorized data user	×	×	×	×
Shen et al. [2]	Authorized data user	×	×	×	✓
Li et al. [3]	Authorized data user	×	×	×	×
Li et al. [4]	Authorized data user + decryption service provider	×	×	×	×
Ours	Data owner	✓	✓	✓	✓

We present a system for data sharing in the cloud computing setting called Labrador (LB). Labrador is the first solution that (i) prevents potential dataset-level leakage after data revealing, (ii) requires no ciphertext or key updating, (iii) provides an incentive to data owners to encourage data sharing, and (iv) ensures auditability. See Table 1 for feature comparison with different state-of-the-art paradigms.

2 Related work

Seminal studies related to this paper include verifiable computation and data sharing.

Verifiable computation. In data analytics, verifying whether a result is correctly computed is important. The existing research on enabling verification can be categorized as either cryptography or replication methods. Cryptography methods [18–25] to ensure the correctness of computation have been studied for a long time. However, cryptography methods are usually too heavy to be applicable.

Replication methods [26–29] use duplicated parties to guarantee consistency in dependable systems. Assuming that the analytic result can be reproduced with identical parameters, one can compare the results, and the correctness is proved if all results are identical. This process is the insight behind [30], in which Dong et al. proposed a replication method to verify outsourced computations. Their construction is based on a rational adversary model. However, in [30], the researcher simply considered the comparison between plaintexts. Note that the analytic result in our setting is encrypted with the HE key held by a data owner; thus, additional steps are needed to enable the comparison between the two HE ciphertexts.

Traditional data sharing. Data sharing [1–6] enables a data owner to share private data with an authorized group. One of the most challenging problems in traditional paradigms is how to ensure data privacy against revoked users.

There are few state-of-the-art works focusing on data sharing. For example, Xu et al. [1] designed a new revocable attribute-based encryption (RABE) scheme to update the stored ciphertext. Utilizing ciphertext updating, revoked users are prevented from recovering the data afterward. In [2], Shen et al. realized key agreement in data sharing using asymmetric balanced incomplete block design (SBIBD). Data access requests from revoked users are prevented by the group manager. In [3], Li et al. introduced an online/offline attribute-based encryption (ABE) scheme to achieve efficiency. A non-revoked user with the desired access structure can successfully decrypt the encrypted data with her secret key. In [4], Li et al. introduced a variant of ABE called LDSS-CP-ABE, where a data user with access privilege can query the cloud server for shared data. User revocation is achieved by lazy re-encryption.

We note that potential dataset-level leakage exists in prior studies since the shared data are trivially revealed to delegated parties. The full knowledge of the shared data is obtained by the adversarial party once these data have been accessed and copied. How to prevent future misuse (e.g., unauthorized republishing) of shared data after the data are revealed to delegated parties has not yet been studied.

Blockchain-based data sharing. In some current studies [31,32], the blockchain protocol is utilized to achieve access control in data sharing. Despite additional auditability, they still suffer from dataset-level leakage and cannot realize long-term data privacy.

3 Preliminaries

3.1 Building blocks

In this subsection, we introduce the cryptographic building blocks used to construct our system. In particular, homomorphic encryption is used to enable privacy-preserving data analytic tasks and sym-

metric encryption is for secure data transfer. As for smart contracts, it is used to (i) store necessary protocol transcripts and (ii) automatically provide rewards to honest parties as specified by the protocol. A straightforward benefit brought about by smart contracts is the auditability of our system.

3.1.1 *Homomorphic encryption*

A homomorphic public key encryption scheme $\mathcal{E}$ can be defined as a quadruple $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$. Let λ denote the security parameter and $\mathcal{F}_{\mathcal{E}}$ be the set of circuits supported by $\mathcal{E}$. Four algorithms incorporated in $\mathcal{E}$ can be defined as follows.

• Key generation. The algorithm $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ outputs a public encryption key pk and a secret decryption key sk when given λ.

• Encryption. The algorithm $c \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, m)$ outputs the ciphertext c according to the input public encryption key pk and plaintext m. For simplicity, $c \leftarrow \mathsf{E}^{\mathcal{E}}_{\mathsf{pk}}(m)$ is adopted to denote the encryption algorithm.

• Decryption. The algorithm $m' \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, c)$ outputs the plaintext m' according to the input secret decryption key sk and ciphertext c. Similar to encryption, $m' \leftarrow \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}}(c)$ is short for decryption.

• Evaluation. The algorithm $c' \leftarrow \mathcal{E}.\mathsf{Eval}(\mathsf{pk}, f, c)$ takes public encryption key pk, a circuit $f \in \mathcal{F}_{\mathcal{E}}$ and ciphertext c encrypted with pk and outputs the c' satisfying $f(\mathsf{D}^{\mathcal{E}}_{\mathsf{sk}}(c)) = \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}}(c')$, in which sk is the secret decryption key related to pk. The property that

$$c' = \mathsf{E}^{\mathcal{E}}_{\mathsf{pk}}(f(m)) = f_{\oplus}(\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}}(m)) \tag{1}$$

holds for HE scheme $\mathcal{E}$, in which $f_{\oplus}$ is the arithmetic function over the ciphertext according to f. In other words, we can compute any circuit $f_{\oplus}$ related to $f \in \mathcal{F}_{\mathcal{E}}$ on a ciphertext encrypted with $\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}}$. Then, using $\mathsf{D}^{\mathcal{E}}_{\mathsf{sk}}$, $f_{\oplus}$ can turn into f on the related plaintext. In the rest of this paper, we will simply use the property of (1), without explicitly calling $\mathcal{E}.\mathsf{Eval}$.

3.1.2 *Asymmetric encryption*

An asymmetric encryption scheme $\mathcal{E}_0 = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

• Key generation. The algorithm $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{E}_0.\mathsf{KeyGen}(1^\lambda)$ outputs a public encryption key pk and a secret decryption key sk while given λ.

• Encryption. The algorithm $c \leftarrow \mathcal{E}_0.\mathsf{Enc}(\mathsf{pk}, m)$ outputs the ciphertext c according to the input public encryption key pk and plaintext m. For simplicity, $c \leftarrow \mathsf{E}^{\mathcal{E}_0}_{\mathsf{pk}}(m)$ is adopted to denote the encryption algorithm.

• Decryption. The algorithm $m' \leftarrow \mathcal{E}_0.\mathsf{Dec}(\mathsf{sk}, c)$ outputs the plaintext m' according to the input secret decryption key sk and ciphertext c. Similar to encryption, $m' \leftarrow \mathsf{D}^{\mathcal{E}_0}_{\mathsf{sk}}(c)$ is short for decryption.

3.1.3 *Smart contract*

Labrador is based mainly on the architecture of Ethereum. Concretely, a smart contract $\mathcal{C}$ is a piece of code deployed on a blockchain, and a smart contract user $\mathcal{U}$ can be specified by its account address $\mathsf{adr}_{\mathcal{U}}$. It can be regarded as trusted.

3.2 System model

In this subsection we describe the entities involved in our system and the formal model of analytic tasks proposed by a data analyst.

3.2.1 *Entities*

Labrador involves six entities: a data analyst, two cloud servers, a data owner, an LB contract and an LB server. Note that the two cloud servers are randomly chosen by the data analyst from all clouds that hold the shared data. The high-level architecture of LB is given in Figure 1. Details about these entities are given below.

• Data analyst $\mathcal{U}$. The data analyst is the entity who desires to analyze the data shared by the data owner and initiates a request for analytic service. The data analyst holds an asymmetric key pair $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathcal{E}_0.\mathsf{KeyGen}(1^\lambda)$ and the account address $\mathsf{adr}_{\mathcal{U}}$ on the blockchain.

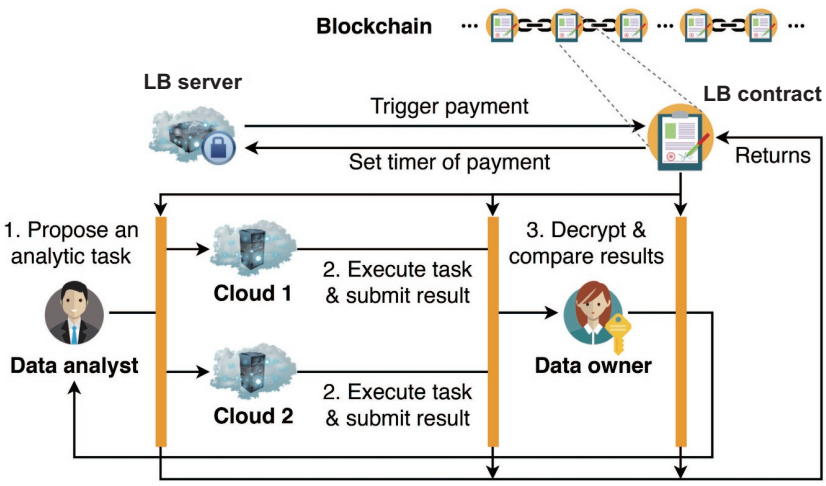

Figure 1 (Color online) High-level architecture of Labrador.

• Cloud server $\mathcal{C}_i$ $(i = 1, 2)$. The cloud server provides storage service for the data owner and analytic service for the data analyst. The two cloud servers aim to earn service fees from the data analyst by enforcing the specified analytic task. Cloud server $\mathcal{C}_i$ has the account address $\mathsf{adr}_{\mathcal{C}_i}$ on the blockchain and an asymmetric key pair $(\mathsf{pk}_{\mathcal{C}_i}, \mathsf{sk}_{\mathcal{C}_i}) \leftarrow \mathcal{E}_0.\mathsf{KeyGen}(1^\lambda)$.

• Data owner $\mathcal{O}$. The data owner is willing to share data with the data analyst if a fair reward is obtained from data sharing and if privacy is protected during data analytics. The data owner holds a homomorphic key pair $(\mathsf{pk}_{\mathcal{O}}, \mathsf{sk}_{\mathcal{O}}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$. Owing to limited resources, the data owner will upload its homomorphically encrypted data to cloud servers. Additionally, account address $\mathsf{adr}_{\mathcal{O}}$ is reserved for $\mathcal{O}$.

• LB contracts $\mathcal{C}_{\mathsf{LB}}$ and LB servers $\mathcal{S}_{\mathsf{LB}}$. These two entities work cooperatively as the core of LB. In detail, the LB contract provides interfaces for those who want to join LB. The LB server is regarded as a trusted third party (TTP) that behaves as a functional extension of the LB contract. These components can be viewed as the same entity with identical account address $\mathsf{adr}_{\mathsf{LB}}$ on the blockchain, while the LB server, as a TTP, holds $(\mathsf{pk}_{\mathsf{LB}}, \mathsf{sk}_{\mathsf{LB}}) \leftarrow \mathcal{E}_0.\mathsf{KeyGen}(1^\lambda)$.

3.2.2 *Analytic task*

For simplicity, assume that data owner $\mathcal{O}$ has previously uploaded its data to all cloud servers available in LB (including $\mathcal{C}_1$ and $\mathcal{C}_2$). Let $\mathsf{data}_{\mathcal{O}}$ denote the data to be shared by $\mathcal{O}$ in plaintext and $\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\mathsf{data}_{\mathcal{O}})$ be the homomorphic ciphertext uploaded.

LB takes $\mathsf{adr}_{\mathcal{O}}$ and arithmetic analytic task $\phi \in \mathcal{F}_{\mathcal{E}}$ as inputs. Formally, $\phi : \mathbb{F}_p^d \to \mathbb{F}_p$, in which the input size d varies according to the desired dataset to use. $\mathsf{adr}_{\mathcal{O}}$ is used to look up $\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\mathsf{data}_{\mathcal{O}})$ in the cloud storage. We use $\phi_{\oplus}$ to denote the arithmetic function over the homomorphic ciphertext according to ϕ. After finishing ϕ, LB outputs the analytic result $\mu = \phi(\mathsf{data}_{\mathcal{O}})$. Implicitly, we assume that (i) $\mu \in \mathbb{F}_p$ can be reproduced by conducting the same analytic task ϕ with the same parameters, (ii) addition $+$ and multiplication $\times$ over $\mathbb{F}_p$ satisfy $+, \times \in \mathcal{F}_{\mathcal{E}}$, and (iii) $\mu^2 - 1 \neq_p 0$, where p is a prime number in size $\mathsf{poly}(\lambda)$ and $\neq_p$ denotes the inequality over $\mathbb{F}_p$ (similarly, $=_p$ for equivalence over $\mathbb{F}_p$). We denote the addition and multiplication over homomorphic ciphertext by $\oplus$ and $\otimes$, respectively. Let $\mathsf{H}(\cdot)$ denote a secure collision-resistant hash function.

3.3 Security assumptions

The following security assumptions are assumed to hold.

• Analytic task. We assume that the analytic task requested by the data analyst is not time-critical; that is, the required time for message delivery in LB is larger than the inherent latency of the low-level blockchain. Furthermore, the results of the same analytic task with identical parameters are required to be identical in plaintext. Moreover, we assume the analytic task is nontrivial; that is, it is difficult for the data analyst to compute ϕ^{-1} with the output of ϕ.

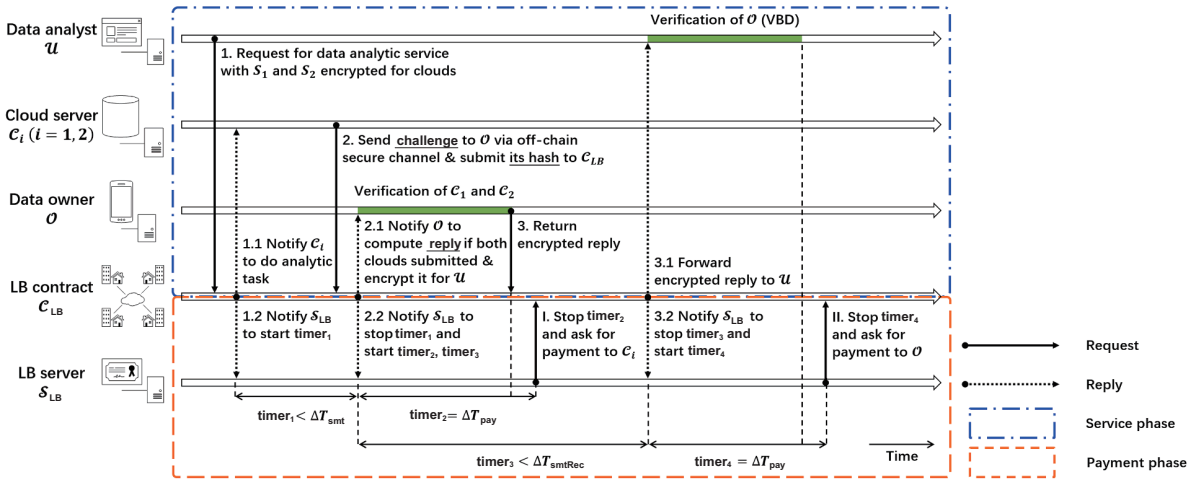

Figure 2 (Color online) Construction of Labrador, which consists of a service phase and payment phase. Note that the payments enforced in the payment phase are triggered by timers (see Subsection 5.1).

• Participant entities. Nontrivially, we assume that all entities involved in LB have different identities. We stress that all participant entities in LB are supposed to be rational in the sense of game theory; however, they can all be malicious. Moreover, no collusive behavior exists among participants. This assumption aims to prevent (i) the common interest (i.e., positively correlated utility functions) of some participants and (ii) a crafted false analytic result agreed to by two cloud servers. To simplify the utility function, we assume that these entities are risk neutral.

• LB server. We assume that $\mathcal{S}_{\mathsf{LB}}$ is a TTP. However, except being an entity to trigger due payments, $\mathcal{S}_{\mathsf{LB}}$ is disengaged in most cases and will be awake only when disputes between entities occur. We view $\mathcal{S}_{\mathsf{LB}}$ as a deterrence to prevent potential cheating in LB. Most of the time, $\mathcal{S}_{\mathsf{LB}}$ works as a relative time counter whenever $\mathcal{U}$ asks for LB service through $\mathcal{C}_{\mathsf{LB}}$.

• Data reliability. The data uploaded to all cloud servers originate from the behavior of a data owner in the real world instead of being crafted or tampered with by cloud servers.

• Computational overhead. The computational overhead for both cloud servers to launch the same analytic task is assumed to be identical. Additional, making up plausible results has a low cost.

• Off-chain channel. We assume the existence of an off-chain channel for secure and authenticated communication between any two blockchain peers.

4 Verifiable blind decryption over HE

Consider the following situation: Alice performed some arithmetic operations on the homomorphically encrypted data shared by Bob. She desires the analytic result in plaintext. Since only Bob has the key to decrypt this ciphertext result, Alice has to turn to Bob for help. But how can Alice be convinced that Bob performed the homomorphic decryption over the ciphertext honestly while keeping the plaintext result unknown to Bob?

To address the aforementioned problem, we propose a novel VBD technique over an HE scheme. The verifiability of VBD comes from the property of over-determined systems, while its blindness relies on an under-determined system. Importantly, this technique is constructed over a (linearly) HE scheme; we use VBD for short in the rest of this paper.

Assume that the HE key pair of Bob is $(\mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{B})$ and that the ciphertext result is $\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_\mathsf{B}}(\mu)$. The finite field $\mathbb{F}_p$ and the parameters of $\mathcal{E}$ are public to both Alice and Bob. Honest Alice first samples $S_1, S_2 \xleftarrow{\$} \mathbb{F}_p$. Then, Alice computes the challenge (b_1, b_2) and sends it to Bob, in which

$$\begin{cases} b_1 = S_1 \cdot \mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_\mathsf{B}}(\mu) \oplus \mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_\mathsf{B}}(S_2), \\ b_2 = S_2 \cdot \mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_\mathsf{B}}(\mu) \oplus \mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_\mathsf{B}}(S_1). \end{cases} \tag{2}$$

Upon receiving (b_1, b_2), Bob simply tries to decrypt it with sk_B. If honest, Bob should obtain the reply

(B_1, B_2) and send it back to Alice, in which

$$\begin{cases} B_1 = S_1 \cdot \mu + S_2, \\ B_2 = S_2 \cdot \mu + S_1. \end{cases} \tag{3}$$

With (B_1, B_2), Alice finally verifies the honesty of Bob. Note that the equalities in system (3) are derived from the additional property of HE.

On the one hand, to see whether system (3) is over-determined for Alice, one can check the number of equations and unknowns in system (3). Clearly, there are two equations with only one unknown (i.e., μ) for Alice; that is, Alice can solve μ from either the first or the second equation in system (3). Let μ_1 be the μ solved from B_1 and μ_2 be that solved from B_2. It follows that $\mu_1 = \mu_2$ if Bob launched decryption honestly. This observation lays the foundation of the verifiability of our VBD technique.

On the other hand, system (3) is under-determined for Bob since there are three unknowns (i.e., S_1, S_2 and μ). We prove in Subsection 6.3 that the excessive number of unknowns guarantees that Bob cannot solve μ with only the knowledge of the reply; that is, Bob is blinded while he decrypts the ciphertext result.

5 System design

5.1 Overview

The insight behind our VBD technique introduced in Section 4 is leveraged in the construction of LB. The tasks of Alice are shared by the data analyst and the two cloud servers, while the data owner is played by Bob. Specifically, S_1 and S_2 are sampled by the data analyst, and each cloud server computes the challenge. The data owner then computes two replies based on the two challenges. If both clouds are honest, the two replies should be identical in plaintext. This process is used to verify the honesty of the clouds. Finally, the identical reply is forwarded to the data analyst who uses our VBD technique to verify whether the data owner honestly decrypts the challenge.

The construction of LB, shown in Figure 2, comprises two phases: service phase and payment phase. The two phases are summarized as follows.

• Service phase. In the service phase, contract calls are organized in the "request-reply" style. Roughly speaking, the service phase is designed to complete the arithmetic task ϕ while ensuring the correctness of μ by introducing two verification processes.

• Payment phase. Given that all results computed in LB should be submitted in time, let ΔT_{smt} denote the time limit for clouds to submit their challenges and $\Delta T_{\mathsf{smtRec}}$ denote the time limit for the reply of the data owner. These time intervals are counted by the LB server, which will notify the LB contract to abort without enforcing payment if timeout occurs. If ϕ finishes without timeout and accusation (see Subsection 5.3), after counting ΔT_{pay} from the time of receiving successful submissions, the LB server will trigger payment by contract call.

Since the idea behind our payment mechanism is easy to understand, this paper mainly discusses how the service phase works in Subsection 5.2. In addition, we describe our game theory technique used to deal with dishonest servers in Subsection 5.3.

5.2 Service phase

5.2.1 *Computation of the cloud server*

With S_1 and S_2 given by the data analyst and the analytic result $\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\mu)$ computed by themselves, the two cloud servers can compute their challenges according to system (2). Note that the HE key pair here is $(\mathsf{pk}_{\mathcal{O}}, \mathsf{sk}_{\mathcal{O}})$ and $\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\mu) = \phi_{\oplus}(\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\mathsf{data}_{\mathcal{O}})) = \mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\phi(\mathsf{data}_{\mathcal{O}}))$. Then, each cloud server forwards its challenge to the data owner using an off-chain secure channel while submitting the hash pair $(\mathsf{H}(b_1), \mathsf{H}(b_2))$ to the LB contract.

Let $\mathsf{CofCloud}$ denote the computation task for each cloud server specified by the data analyst. We provide the details in Algorithm 1.

Algorithm 1 CofCloud

Input: $\mathsf{adr}_{\mathcal{O}}$: account address of data owner; ϕ: analytic task; (S_1, S_2): random values;
Output: (b_1, b_2): the challenge;
1: Look up $\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\mathsf{data}_{\mathcal{O}})$ and $\mathsf{pk}_{\mathcal{O}}$ in storage using $\mathsf{adr}_{\mathcal{O}}$;
2: Translate ϕ into $\phi_{\oplus}$;
3: $x = \phi_{\oplus}(\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\mathsf{data}_{\mathcal{O}}))$;
4: $b_1 = S_1 \cdot x \oplus \mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(S_2)$, $b_2 = S_2 \cdot x \oplus \mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(S_1)$;
5: **return** (b_1, b_2).

Algorithm 2 VofClouds

Input: (b_{11}, b_{12}): challenge from cloud 1; (b_{21}, b_{22}): challenge from cloud 2; $\mathsf{sk}_{\mathcal{O}}$: homomorphic secret key of data owner; $\mathsf{pk}_{\mathcal{U}}$: asymmetric public key of data analyst;
Output: $(\mathsf{isValid}, V_1, V_2)$: the encrypted reply;
1: **if** $\mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{11}) = \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{21})$ **and** $\mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{12}) = \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{22})$ **then**
2: $B_1 = \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{11})$, $B_2 = \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{12})$;
3: **return** $(\mathbf{true}, \mathsf{E}^{\mathcal{E}_0}_{\mathsf{pk}_{\mathcal{U}}}(B_1), \mathsf{E}^{\mathcal{E}_0}_{\mathsf{pk}_{\mathcal{U}}}(B_2))$;
4: **else**
5: **return** $(\mathbf{false}, \perp, \perp)$;
6: **end if**

5.2.2 *Verification of cloud servers*

Upon receiving challenges from both specified cloud servers, the data owner is supposed to verify whether the cloud servers have computed $\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\mu)$ honestly. This verification is achieved simply by decrypting the two challenges and comparing the two derived replies. Note that the reply for VBD is obtained simultaneously in this procedure.

Let $\mathsf{VofClouds}$ be the verification done by the data owner and $(b_{i1}, b_{i2}) \leftarrow \mathcal{C}_i.\mathsf{CofCloud}(\mathsf{adr}_{\mathcal{O}}, \phi, S_1, S_2)$ $(i = 1, 2)$. Two possible cases can occur.

• $\mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{11}) = \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{21})$ and $\mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{12}) = \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{22})$. In this case, it is safe to assert with a high probability that both cloud servers are honest according to our security assumptions. The data owner will then encrypt the identical reply using the asymmetric public key $\mathsf{pk}_{\mathcal{U}}$ to secretly forward it to the data analyst.

• $\mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{11}) \neq \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{21})$ or $\mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{12}) \neq \mathsf{D}^{\mathcal{E}}_{\mathsf{sk}_{\mathcal{O}}}(b_{22})$. In contrast to the first case, at least one cloud server is cheating. The data owner will raise an accusation against the clouds and call the LB sever to address this dispute.

The procedure described above is presented in Algorithm 2. The data owner will forward the encrypted reply to the data analyst.

5.2.3 *Verification of data owner*

The data analyst will verify the correctness of the decryption by the data owner using VBD. More precisely, the data analyst first performs asymmetric decryption to obtain the reply (B_1, B_2) in plaintext. Then, with the knowledge of S_1 and S_2, the data analyst computes

$$\begin{cases} \mu_1 =_p S_1^{-1} \cdot (B_1 - S_2), \\ \mu_2 =_p S_2^{-1} \cdot (B_2 - S_1). \end{cases} \tag{4}$$

If the data owner decrypts the challenge honestly, $\mu_1 = \mu_2$; that is, whenever $\mu_1 \neq \mu_2$ is observed, the data analyst can accuse the data owner of deviating from the protocol and thus refuse to pay the reward. Otherwise, the data owner is honest and should be fairly rewarded. Algorithm 3 presents the verification procedure enforced by the data analyst.

5.3 Deterrence in Labrador

5.3.1 *Global deposit*

To ensure fair payment, a data owner or a cloud server has to pay a deposit before joining LB, which is called the global deposit. For simplicity, we assume the global deposit of each entity is always sufficient during the system execution. Once a dishonest entity is spotted, a forfeit will be deducted from its global deposit.

Algorithm 3 VofOwner

Input: (V_1, V_2): encrypted reply; $\mathsf{sk}_{\mathcal{U}}$: asymmetric secret key; (S_1, S_2): random values;
Output: $(\mathsf{isValid}, \mu)$: the analytic result;
1: $B_1 = \mathsf{D}^{\mathcal{E}_0}_{\mathsf{sk}_{\mathcal{U}}}(V_1)$, $B_2 = \mathsf{D}^{\mathcal{E}_0}_{\mathsf{sk}_{\mathcal{U}}}(V_2)$;
2: $\mu_1 =_p S_1^{-1} \cdot (B_1 - S_2)$, $\mu_2 =_p S_2^{-1} \cdot (B_2 - S_1)$;
3: **if** $\mu_1 = \mu_2$ **then**
4: **return** $(\mathbf{true}, \mu_1)$;
5: **else**
6: **return** $(\mathbf{false}, \perp)$;
7: **end if**

5.3.2 *Accusation*

Accusation is incorporated into LB to reveal dishonest behavior. LB provides AccuseCloud for the data owner and AccuseOwner for the data analyst to raise an accusation of dishonesty of the clouds and data owner, respectively. A well-grounded accusation will lead to a forfeit being deducted from the cheater's global deposit. The forfeited deposit will be used for (i) compensation for other honest entities, (ii) reward for the entity who makes the accusation, and (iii) the fee for the LB server to validate this accusation.

5.3.3 *Accusation validation by the LB server*

The LB server is responsible for validating accusations. Consider the following two cases.

• AccuseCloud is called. In this case, the LB server will fetch $\mathsf{E}^{\mathcal{E}}_{\mathsf{pk}_{\mathcal{O}}}(\mathsf{data}_{\mathcal{O}})$, S_1 and S_2 from both cloud servers and $\mathsf{sk}_{\mathcal{O}}$ from the data owner. With these data, the LB server will redo CofCloud and VofClouds to identify the cheating cloud(s) or data owner.

• AccuseOwner is called. In this case, the LB server will redo VofOwner. The accusation will be regarded as well grounded (i.e., the data owner is dishonest) if VofOwner called by LB server outputs $(\text{false}, \perp)$ and vice versa.

5.3.4 *Cost, reward and forfeit*

LB contract provides a fair payment functionality for LB participants. We use the notation $\$\_$ to denote a possible monetary variable. We list all variables in LB as follows, and we implicitly assume that all the variables are positive and counted in the same unit (e.g., ether). More specifically, $\$\mathsf{f}$ denotes the fees paid by the data analyst to initiate a new request, from which the rewards for other entities, as well as the forfeit, are deducted; $\$\mathsf{G_C}$, $\$\mathsf{G_O}$ denote the least required amount of global deposit for a cloud server and a data owner to join LB, respectively; $\$\mathsf{c_C}$, $\$\mathsf{c_O}$ denote the cost for a cloud server to correctly compute ϕ and for a data owner to do VofClouds, respectively; $\$\mathsf{v}$ denotes the reward for LB server to validate an accusation; $\$\mathsf{r_C}$, $\$\mathsf{r_O}$ denote the rewards for a cloud server and a data owner to honestly perform their instructed tasks, respectively; $\$\mathsf{d}$ denotes the forfeited deposit that is deducted from the global deposit of a dishonest entity; $\$\mathsf{a}$ denotes the reward for a data owner to raise a well-grounded accusation; $\$\mathsf{r_{LB}}$ denotes the reward for LB contract and LB server to process a LB service request that terminates normally.

In addition, the following relationships among these variables are supposed to hold:

• $\$\mathsf{f} > \$\mathsf{d} + \$\mathsf{v} + 2\$\mathsf{r_C} + \$\mathsf{r_O}$: this guarantees that a data analyst provides a sufficient service fee to initiate a new LB service transaction.

• $\$\mathsf{G_C} > \$\mathsf{d}$ and $\$\mathsf{G_O} > \$\mathsf{d}$: this guarantees that the deposits of a cloud server and a data owner are sufficient to be forfeited.

• $\$\mathsf{r_C} > \$\mathsf{c_C}$ and $\$\mathsf{r_O} > \$\mathsf{c_O}$: this prevents a cloud server and a data owner from suffering deficits in computational cost.

• $\$\mathsf{v} > 2\$\mathsf{r_C}$: this guarantees that it makes sense to lower the computational cost by using two cloud servers instead of using a TTP.

• $\$\mathsf{d} > \$\mathsf{v} + \$\mathsf{a} + 2\$\mathsf{r_C} + \$\mathsf{r_O}$: this guarantees that a data analyst will not suffer from any pecuniary loss if abortion led by others occurs.

• $\$\mathsf{v} > \$\mathsf{r_{LB}}$: this guarantees that the LB server will not be used to validate an accusation in most cases given its higher cost.

6 Security analysis

In this section, we present the security analysis of LB. Recall that we assume that any subset of participants is not in collusion in Subsection 3.3. We first show that our VBD technique achieves both verifiability and blindness (Theorem 2 and Theorem 1 respectively). Then, based on the security of our VBD technique, both the fairness and the honest behavior of involved entities can be guaranteed (Theorem 3). In addition, we show that long-term privacy can also be achieved (Theorem 4).

6.1 Verifiability and blindness of VBD

Recall that in LB construction, the data analyst and the two clouds share the role of Alice, while the data owner plays the role of Bob (see Section 4 and Subsection 5.1). Indeed, the two clouds cannot decrypt the asymmetrically encrypted reply returned by Algorithm 2; they just participate in generating the challenges. In the following two theorems, we will show that if the challenges are generated honestly by the two clouds, both verifiability and blindness of the data owner are achieved. We start from proving blindness, which lays the foundation of verifiability.

Theorem 1 (Blindness of VBD)**.** A data owner $\mathcal{O}$ who knows only B_1, B_2 in system (3) cannot acquire a unique solution (μ, S_1, S_2); that is, $\mathcal{O}$ cannot learn the analytic result μ to be decrypted.

Proof. Let $\mu = k$ ($k \in \mathbb{F}_p, k^2 - 1 \neq_p 0$); then, system (3) can be rewritten as

$$\begin{cases} kS_1 + S_2 =_p B_1, \\ kS_2 + S_1 =_p B_2. \end{cases} \tag{5}$$

Note that p is a prime number and any nonzero element in $\mathbb{F}_p$ has a unique modular multiplicative inverse. From system (5), we know that

$$\begin{cases} S_1 =_p (k^2 - 1)^{-1}(kB_1 - B_2), \\ S_2 =_p (k^2 - 1)^{-1}(kB_2 - B_1), \end{cases} \tag{6}$$

in which $(k^2 - 1)^{-1}$ is the modular multiplicative inverse of $(k^2 - 1)$ under $\mathbb{F}_p$ when $(k^2 - 1) \neq_p 0$. System (6) implies that the solution (μ, S_1, S_2) of system (3) is dependent on μ.

Since p is a prime and the quadratic residue $k^2 =_p 1$ will have no more than two solutions, the solution space of system (6) is of size $\Theta(|\mathbb{F}_p|) = 2^{\mathsf{poly}(\lambda)}$. Exhaustive search can succeed with only negligible probability. Provided that S_1 and S_2 are both used once, no information about the analytic result is revealed.

Theorem 2 (Verifiability of VBD)**.** Assume the honesty of both cloud servers. Then, the data owner $\mathcal{O}$ is said to correctly decrypt the challenge in step 2 of VofClouds iff $\mu_1 = \mu_2$ holds, in which μ_1 and μ_2 are computed by the data analyst $\mathcal{U}$ according to system (4).

Proof. On the one hand, it is straightforward to see that $\mu_1 = \mu_2$ holds if $\mathcal{O}$ honestly decrypts the challenge.

On the other hand, to argue that $\mu_1 = \mu_2$ holds only if the decryption is conducted honestly, we might as well assume that $\mu_1 = \mu_2$ still holds for dishonest $\mathcal{O}$, i.e., $\mathcal{O}$ passes VofOwner even if it is dishonest. This assumption implies that the data owner is able to construct a $\mu' \neq \mu$ that makes $S_1' =_p (\mu'^2 - 1)^{-1}(\mu' B_1 - B_2)$ and $S_2' =_p (\mu'^2 - 1)^{-1}(\mu' B_2 - B_1)$. From the perspective of $\mathcal{U}$, since μ' passed VofOwner carried out by itself, (μ', S_1, S_2) must be the solution obtained from system (3) with the knowledge of S_1 and S_2. In other words, (μ', S_1, S_2) and (μ', S_1', S_2') are both the solution of system (3), and, essentially, $S_1 = S_1'$ and $S_2 = S_2'$ by applying system (6) to the same μ'. This result implies that $\mathcal{O}$ knows S_1 and S_2. With the knowledge of (S_1, S_2, B_1, B_2), $\mathcal{O}$ can learn μ from system (3), which is contradictory to Theorem 1.

6.2 Security of Labrador

Similar to [30], we define our security goals under the rational adversary model. Recall that all participants in LB may cheat for the following motivations:

• A dishonest cloud server may wish to (i) receive payment with little effort spent to perform the analytic task (i.e., generating its challenge) or (ii) prevent the data analyst from learning the result.

• A dishonest data owner may wish to (i) receive payment with little effort spent to perform the decryption task (i.e., generating its reply) or (ii) prevent the data analyst from learning the result.

• A dishonest data analyst may wish to deny the promised rewards after learning the result.

Hence, we model the execution of LB as a game $\mathcal{G}$ in the sense of game theory; that is, in LB, one participant may decide whether or not to be an honest participant according to the others' strategies. For instance, a cloud server may decide to behave honestly, knowing that the data owner will honestly perform VofClouds. We analyze the game $\mathcal{G}$ under the non-collusion assumption. Two security goals are defined as follows.

Definition 1 (Fairness)**.** Let Δ_{P} denote the deposit difference of some participant P after each round of LB execution. For an honest P, except the data analyst $\mathcal{U}$, $\Delta_{\mathsf{P}} > 0$ is always satisfied, while $\Delta_{\mathsf{P}} < 0$ for dishonest P. For the data analyst $\mathcal{U}$, if honest, $\Delta_{\mathcal{U}} \geqslant -(2\$\mathsf{r}_{\mathsf{C}} + \$\mathsf{r}_{\mathsf{O}} + \$\mathsf{r}_{\mathsf{LB}})$; otherwise, $\Delta_{\mathcal{U}} < -(2\$\mathsf{r}_{\mathsf{C}} + \$\mathsf{r}_{\mathsf{O}} + \$\mathsf{r}_{\mathsf{LB}})$.

Definition 2 (Honesty)**.** In each round of LB execution, the two cloud servers, the data owner and the data analyst will behave honestly as instructed by LB.

Definition 1 aims to capture the feature that, in LB, an honest participant will not suffer from financial loss. In other words, dishonest parties compensate for honest parties. Based on the fairness feature, honesty is achieved by enforcing every participant not to deviate from the protocol to reach their highest payoffs, which is informally defined in Definition 2.

Theorem 3. Assume the rationality of all participants ($\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{O}$ and $\mathcal{U}$) and the non-collusion setting. Then, LB achieves both fairness and honesty according to Definitions 1 and 2.

Proof. (Sketch) According to the deterrence mechanisms in Subsection 5.3, it is easy to see that fairness is achieved. To prove honesty in LB, we can follow the example of [30] and prove that the LB game $\mathcal{G}$ converges to a sequential equilibrium [33] where all participants behave honestly to achieve their highest payoffs. In particular, in the assumed non-collusion setting, the two clouds behave as in the prisoner's contract of [30]. Now we introduce the data analyst and the data owner to the original prisoner game. The critical thing is that the data owner can choose whether or not to expose the dishonesty of clouds. However, the choice of the data owner depends on whether or not the data analyst would deny the rewards after learning the result. With the help of the deterrence mechanisms in Subsection 5.3, we can conclude that the data analyst has to be honest if she wants to achieve her highest payoffs. The data owner would be honest for the same reason. The remaining part of this proof is similar to that in [30].

6.3 Long-term privacy

Theorem 4 (Confidentiality of μ)**.** Assume there is a secure HE scheme $\mathcal{E}$ and a secure asymmetric encryption scheme $\mathcal{E}_0$. In the non-collusion setting, LB achieves the confidentiality of μ against everyone except the data analyst $\mathcal{U}$ with overwhelming possibility.

Proof. If the cloud server $\mathcal{C}_i$ can obtain the value of μ, then the server can secretly carry out $\phi : \phi(x) = x$ without any request from $\mathcal{C}_{\mathsf{LB}}$ to lead to $\mu = \mathsf{data}_{\mathcal{O}}$. In other words, $\mathcal{C}_i$ can arbitrarily recover the plaintext of $\mathcal{O}$'s data without homomorphic decryption, which implies that $\mathcal{C}_i$ can break $\mathcal{E}$. Note that the trivial ϕ satisfies $\phi \in \mathcal{F}_{\mathcal{E}}$. In addition, we learn from Theorem 1 that the data owner $\mathcal{O}$ cannot recover μ. For the outside of the protocol execution, the observed information includes (i) the hashes of the two homomorphic ciphertexts of B_1 and B_2 (i.e., $\mathsf{H}(b_{i1})$ and $\mathsf{H}(b_{i2})$) and (ii) the asymmetric ciphertext of μ. That is, to obtain the value of μ, the entities either have to compute H^{-1} (then break $\mathcal{E}$ and solve system (3) without S_1 and S_2) or simply to break $\mathcal{E}_0$.

Theorem 4 shows that the analytic result μ is merely leaked to the data analyst. Moreover, only μ, instead of full knowledge of the shared data, is learned by the data analyst in each round of LB execution. We do not consider the situation where the data analyst can recover the whole dataset with a few requests to LB. In such a case, long-term privacy is trivially achieved by merely revealing μ to the data analyst.

7 Evaluations

In this section, we realize a proof-of-concept (PoC) implementation of LB. This PoC implementation merely captures the service phase, working as a subroutine contract of the full version of the LB contract. We instantiate the secure collision-resistant hash function $\mathsf{H}(\cdot)$ with SHA-256. $\mathcal{E}$ is instantiated with the BFV scheme [13] (implemented by Microsoft SEAL v2.3.1) without SIMD optimization, while $\mathcal{E}_0$ is

Table 2 Parameters for evaluating the PoC implementation

Case	HE parameter			λ (bits)	ϕ	Input size d	Storage cost (GB)
	n	Length of p (bits)	t				
#1	4096	109	65407	> 128	dot product	500	0.128
#2						1000	0.256
#3						5000	1.3
#4						10000	2.6
#5	8192	218	65407	> 128		1000	1
#6	16384	438	65407	> 128		1000	4.2
#7	4096	109	65407	> 128	$\phi : \phi(x) = x$	1	–
#8	8192	218	65407	> 128		1	–
#9	16384	438	65407	> 128		1	–
#10	4096	109	65407	> 128	dot product	{500, 1000, 5000, 10000}	–
#11	8192	152	65407	> 192			
#12	8192	118	65407	> 256			

instantiated with RSA-2048. This PoC implementation is realized in Node.js and C++ using the web3.js module to communicate with a low-level Ethereum blockchain. To construct a private network for testing, we have spawned four Go-Ethereum (Geth) nodes (data analyst, data owner, and two cloud servers) in the same physical machine with an i7-7700 3.60 GHz processor and 16 GB memory capacity. Meanwhile, the number of miner threads of each Geth node was set to 2. The end-to-end delay and the time for off-chain data transfer between two nodes were omitted in our experiment. No optimization was adopted in this PoC implementation.

Test cases. The parameters for evaluating the LB in our experiment are listed in Table 2. Note that the parameters in SEAL are the tuple (n, p, t), where n is a power of 2 and determines the polynomial modulus $x^n + 1$, p determines the ciphertext space $\mathbb{Z}_p[x]/(x^n + 1)$, and t determines the plaintext space $\mathbb{Z}_t[x]/(x^n + 1)$. The lengths of the selected p's are given in Table 2. The security level promised by the tuple (n, p, t) is denoted by λ in Table 2. We considered two types of ϕ: (i) the ϕ to compute the dot product of two d-dimensional homomorphically encrypted vectors (in cases #1–#6) and (ii) the trivial ϕ such that $\phi : \phi(x) = x$ (insecure but for benchmark only), which simply returns the dataset in plaintext (in cases #7–#9). Furthermore, to assess the performance of some key operations in LB under different security levels, we consider another set of parameters (#10–#12). In these cases, we merely consider the ϕ that computes the dot product of two homomorphically encrypted vectors. We evaluate two types of key operations: (i) the off-chain computation of ϕ launched by $\mathcal{C}_i$ and (ii) the off-chain re-encryption launched by $\mathcal{O}$ according to steps 2 and 3 in Algorithm 2. Notice, the times for these operations are independent of the performance of the low-level blockchain. Input size d is in $\{500, 1000, 5000, 10000\}$ for each security level.

Performance. We ran 20 rounds for each set of parameters in Table 2 to evaluate the performance of this PoC implementation. Two cloud servers are involved in our construction. Therefore, the maximum time for each server to perform CofCloud (labeled as "max CofCloud") determines the time to fire the event to notify the data owner to conduct VofClouds.

The bottleneck of LB comes from the time for CofCloud. Clearly, Figure 3(a) shows that the maximum time for CofCloud increases as the size of the dataset increases. Moreover, to maintain an approximately identical security level (e.g., >128 bits in cases #2, #5, and #6), the larger n is, the more bits of p are required. Recall that p determines the ciphertext space, which will increase the storage cost of the dataset in the ciphertext. From Figure 3(b), we also know that as n increases, the maximum time for the cloud servers to do CofCloud is the bottleneck of the total time consumed. For instance, in case #6, the average maximum time for CofCloud (353 s) consumes nearly 88.41% of the whole time used (399 s).

In LB, the time for VofClouds and that for VofOwner are sensitive to both the growth of the size of the dataset and the increase of n. Generally, it takes the data owner a few seconds to do VofClouds (< 5 s in our experiments on average), while data analyst spends only a few milliseconds on VofOwner (< 4 ms on average). Note that these overheads are affordable owing to their limited computational ability.

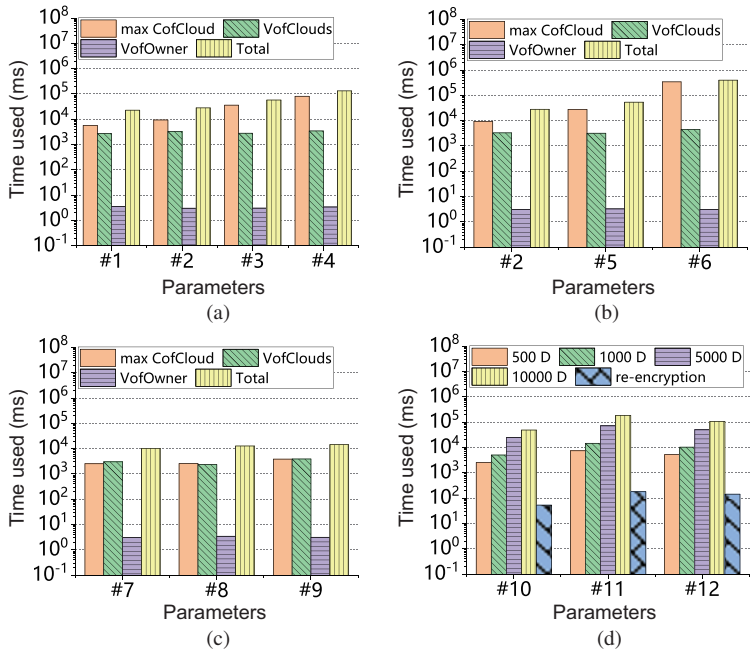

Figure 3 (Color online) Experimental results. Performance of Labrador in calculating (a) the dot product with different input sizes, (b) the dot product with different homomorphic parameters, and (c) the trivial ϕ with different homomorphic parameters. (d) Performance of two off-chain operations with different security levels.

8 Conclusion

Data privacy is a significant principle in data sharing. In this paper, we propose a solution named Labrador for data sharing with guaranteed long-term data privacy, as opposed to the short-term data privacy from the existing methods. In our solution, two cloud servers are utilized to perform and verify the analytic task over a homomorphically encrypted dataset. Using the VBD technique designed in this paper, the data analyst can verify whether the analytic result is correctly decrypted by the data owner. Furthermore, this technique guarantees that the data owner will never learn the analytic result. A smart contract is used to facilitate the incentive mechanism and auditability in our solution, which is helpful to ensure the scalability of the data-sharing community.

We design a game among participants and prove the security of Labrador by assuming malicious but rational participants in the non-collusion setting. We also present an evaluation of our solution, which demonstrates that our method is feasible.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61672300), National Natural Science Foundation of Tianjin (Grant No. 18ZXZNGX00140), National Natural Science Foundation for Outstanding Youth Foundation (Grant No. 61722203), Program for Young Changjiang Scholars in University of Ministry of Education of China, and Natural Science Foundation of China for Joint Fund Project (Grant No. U1936218).

References

1 Xu S M, Yang G M, Mu Y, et al. Secure fine-grained access control and data sharing for dynamic groups in the cloud. IEEE Trans Inform Forensic Secur, 2018, 13: 2101–2113
2 Shen J, Zhou T Q, Chen X F, et al. Anonymous and traceable group data sharing in cloud computing. IEEE Trans Inform Forensic Secur, 2018, 13: 912–925
3 Li J, Zhang Y H, Chen X F, et al. Secure attribute-based data sharing for resource-limited users in cloud computing. Comput Secur, 2018, 72: 1–12
4 Li R X, Shen C L, He H, et al. A lightweight secure data sharing scheme for mobile cloud computing. IEEE Trans Cloud Comput, 2018, 6: 344–357
5 Shao J, Lu R X, Lin X D. Fine-grained data sharing in cloud computing for mobile devices. In: Proceedings of International Conference on Computer Communications, Kowloon, 2015. 2677–2685
6 Yuan J W, Yu S C. Efficient public integrity checking for cloud data sharing with multi-user modification. In: Proceedings of International Conference on Computer Communications, Toronto, 2014. 2121–2129
7 Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theor, 2014, 6: 1–36
8 Smart N P, Vercauteren F. Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Proceedings of Public Key Cryptography, Paris, 2010. 420–443
9 Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (Standard) LWE. In: Proceedings of IEEE 52nd Annual Symposium on Foundations of Computer Science, Palm Springs, 2011. 97–106

10 van Dijk M, Gentry C, Halevi S, et al. Fully homomorphic encryption over the integers. In: Advances in Cryptology—EUROCRYPT 2010. Berlin: Springer, 2010. 24–43
11 Gentry C, Halevi S, Smart N P. Fully homomorphic encryption with polylog overhead. In: Advances in Cryptology—EUROCRYPT 2012. Berlin: Springer, 2012. 465–482
12 Gentry C, Sahai A, Waters B. Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology—CRYPTO 2013. Berlin: Springer, 2013. 75–92
13 Fan J F, Vercauteren F. Somewhat practical fully homomorphic encryption. IACR Cryptol ePrint Archive, 2012. https://eprint.iacr.org/2012/144
14 Liu Y, Luo Y, Zhu Y W, et al. Secure multi-label data classification in cloud by additionally homomorphic encryption. Inf Sci, 2018, 468: 89–102
15 Chen H, Laine K, Rindal P. Fast private set intersection from homomorphic encryption. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, Dallas, 2017. 1243–1255
16 Lu W J, Kawasaki S, Sakuma J. Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. In: Proceedings of the 24th Annual Network and Distributed System Security Symposium, San Diego, 2017
17 Zhang L, Li X-Y, Liu Y H, et al. Verifiable private multi-party computation: ranging and ranking. In: Proceedings of International Conference on Computer Communications, Turin, 2013. 605–609
18 Fiore D, Gennaro R, Pastro V. Efficiently verifiable computation on encrypted data. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, 2014. 844–855
19 Elkhiyaoui K, Önen M, Azraoui M, et al. Efficient techniques for publicly verifiable delegation of computation. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, Xi'an, 2016. 119–128
20 Song W, Wang B, Wang Q, et al. Publicly verifiable computation of polynomials over outsourced data with multiple sources. IEEE Trans Inform Forensic Secur, 2017, 12: 2334–2347
21 Gennaro R, Gentry C, Parno B. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Advances in Cryptology—CRYPTO 2010. Berlin: Springer, 2010. 465–482
22 Backes M, Fiore D, Reischuk R M. Verifiable delegation of computation on outsourced data. In: Proceedings of ACM SIGSAC Conference on Computer & Communications Security, Berlin, 2013. 863–874
23 Parno B, Howell J, Gentry C, et al. Pinocchio: nearly practical verifiable computation. In: Proceedings of IEEE Symposium on Security and Privacy, Berkeley, 2013. 238–252
24 Zhuo G Q, Jia Q, Guo L K, et al. Privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing. In: Proceedings of International Conference on Computer Communications, San Francisco, 2016. 1–9
25 Zheng Q J, Xu S H, Ateniese G. VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: Proceedings of International Conference on Computer Communications, Toronto, 2014. 522–530
26 Canetti R, Riva B, Rothblum G N. Practical delegation of computation using multiple servers. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, Chicago, 2011. 445–454
27 van den Hooff J, Kaashoek M F, Zeldovich N. VerSum: verifiable computations over large public logs. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, 2014. 1304–1316
28 Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery. ACM Trans Comput Syst, 2002, 20: 398–461
29 Cachin C, Kursawe K, Shoup V. Random oracles in Constantinople: practical asynchronous Byzantine agreement using cryptography. J Cryptol, 2005, 18: 219–246
30 Dong C Y, Wang Y L, Aldweesh A, et al. Betrayal, distrust, and rationality: smart counter-collusion contracts for verifiable cloud computing. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, Dallas, 2017. 211–227
31 Xia Q, Sifah E B, Asamoah K O, et al. MeDShare: trust-less medical data sharing among cloud service providers via blockchain. IEEE Access, 2017, 5: 14757–14767
32 Xia Q, Sifah E B, Smahi A, et al. BBDS: blockchain-based data sharing for electronic medical records in cloud environments. Information, 2017, 8: 44
33 Maschler M, Solan E, Zamir S. Game Theory. Cambridge: Cambridge University Press, 2013