PRIDA: PRIvacy-preserving Data Aggregation with multiple data customers

Beyza Bozdemir*
beyza.bozdemir@eurecom.fr
Norton Research Group
EURECOM

Betül Aşkın Özdemir
baskinoz@esat.kuleuven.be
COSIC, KU Leuven
Leuven, Belgium

Melek Önen
melek.onen@eurecom.fr
EURECOM
Sophia Antipolis, France

ABSTRACT

We propose a solution for user privacy-oriented privacy-preserving data aggregation with multiple data customers. Most existing state-of-the-art approaches present too much importance on performance efficiency and seem to ignore privacy properties except for input privacy. Most solutions for data aggregation do not generally discuss the users' birthright, namely their privacy for their own data control and anonymity when they search for something on the browser or volunteer to participate in a survey. Still, they are ambitious to secure data customers' rights (which should come later). They focus on resulting in an efficiency-oriented data aggregation enabling input privacy only. We aim to give importance to user privacy, and we have designed a solution for data aggregation in which we keep efficiency in balance. We show that PRIDA provides a good level of computational and communication complexities and is even better in timing evaluation than existing studies published recently (i.e., Bonawitz et al. (CCS'17), Corrigan-Gibbs et al. (NSDI'17), Bell et al. (CCS'20), Addanki et al. (SCN'22)). We employ threshold homomorphic encryption and secure two-party computation to ensure privacy properties. We balance the trade-off between a proper design for users and the desired privacy and efficiency.

KEYWORDS

Data aggregation, User Privacy, Multiple Data Customers, Threshold Homomorphic Encryption, Secure Two-party Computation, Multi-key Homomorphic Encryption.

1 INTRODUCTION

Data aggregation has become one of the most indispensable tools used by organizations for decision-making, innovation, increasing efficiency, and ultimately driving growth. In order to increase precision and accuracy, companies are tempted to collect, process, and analyze various types of data coming from their users or stakeholders, which often include privacy-sensitive information or business data. If compromised by malicious parties, data stored without proper protection could result in serious privacy and security breaches, maybe even identity theft and fraud. The cost of a data breach, on average, has ramped up to \$4.5M according to the IBM statistics [38]: 51% of organizations plan to increase the privacy and security of the data they analyze, not only to comply with specific privacy-related regulations [2 23 62] but also to avoid these catastrophic events that could result in extreme reputational and monetary damage from happening. Protecting privacy-sensitive data is not just a legal requirement but also a matter of public trust. Conducting data aggregation in a privacy-preserving manner is a way to reduce the risk of privacy breaches and maintain the trust of users and stakeholders. This will help organizations make better decisions, improve their operations, and maintain their reputation in an increasingly data-driven world.

Furthermore, organizations may need domain-specific expertise and/or computational resources to perform data aggregation operations in a privacy-preserving manner. Hence, they usually rely on the existence of third-party cloud servers for such analytics operations to outsource this task. These servers are usually named *aggregators* that offer aggregation operations over data collected from a large number of users. With the existence of this third-party *aggregator*, the implementation of privacy-preserving data aggregation solutions becomes even more essential.

As a reaction to the underlying regulations, low-cost privacy-preserving data aggregation solutions, such as data anonymization [51] and federated learning [36 50 53 56 77 84], were immediately proposed and quickly adopted by organizations. Although data anonymization ensures the privacy of an individual by anonymizing identifiers or by erasing sensitive data, this option is not a foolproof method since [47 70 76 78 80 81] have proven the feasibility of deanonymization. Similarly, FL cannot ensure privacy perfectly: [36 50 53 56 77 84] have unveiled that leaking private data is still feasible in FL. For example, authors have showed that the aggregator could reconstruct the profiles of local training data of a target user by observing the gradient vectors submitted by users, and also, the aggregation server can infer privacy-sensitive properties of local training and conduct membership inference attacks [53 56] by observing gradient or model updates.

An alternative data aggregation solution to protect privacy while still being able to process sensitive data, leverages secure multiparty computation [8 37 69] or homomorphic encryption [21 22 29]. The problem of privacy-preserving data aggregation has intensively been studied, a large corpus of existing work such as [11 25 41 46 55 79 83], but they focus on a simpler scenario where the aggregation operation is not outsourced to a third-party cloud server, and there is **only one stakeholder** interested in this aggregation result: **The aggregator is that interested stakeholder** and can have access to the aggregation result while still not being able to have access to individual data. Few prior solutions like [46] seem to consider a bit more realistic setting whereby the aggregation operation is outsourced to a third-party aggregator with the existence of only one stakeholder interested in the aggregation result. However, these solutions still let the aggregator access the aggregated result. They **violate the result privacy**. Furthermore, a few others could consider having multiple stakeholders who employ the third-party aggregator to access the aggregation result, but they **do not provide any options for the users to choose the stakeholders** who can have access to the result [9 12 13 24 68 72 82] and **to be anonymous** while joining data aggregation with their private data, or they do *not consider the potential collusion among parties*.

1.1 PRIDA: Preview

This paper presents a novel privacy-preserving data aggregation solution called PRIDA that is designed to be suitable for real-world scenarios as enumerated as follows: (i) PRIDA, by privacy-by-design, provides the care to be transparent to the users, we name *data owners* (DOs), who are able to gain the *control their data without being profiled (i.e., being anonymous to stakeholders interested in the aggregation result)* while participating in an aggregation operation. We call this transparency to DOs, **user privacy**. DOs should be free not to share information about themselves as their primary birthright. This crucial birthright should be protected, and DOs should not have to share anything inherently about themselves while doing a simple Google search[1,2]. (ii) PRIDA supports, by design, *multiple stakeholders* that we now name **data customers** (DCs). PRIDA considers user privacy in the presence of multiple data customers. **These two crucial and more real-world use case adaptable privacy properties** have not been studied at the same time by the state-of-the-art solutions. (iii) None DCs and aggregators, **unlike prior solutions**, can access the aggregation result unless multiple DOs allow the corresponding DC to. PRIDA enables the corresponding DCs to obtain an aggregation result, which is private only to them; we call it **result privacy**, as well as provides the traditional and essential privacy goal for DOs, i.e., guaranteeing **input privacy**.

To ensure all these privacy and security goals (See Section 2.1), we combine **threshold homomorphic encryption (Th-FHE)** [4 49], whereby one can perform operations over data encrypted corresponding to multiple keys, with **secure two-party computation (2PC)** [8], where two parties jointly perform a function over private inputs without disclosing them. PRIDA utilizes two aggregators in place of one aggregator used in the more frequently setting of state-of-the-art solutions [7 9–11 13 25 40 68 71 72 79 82 83]. The introduction of this additional aggregator enables DOs to be anonymous to DCs, i.e., DCs are not able to distinguish whether a particular DO has sent their data to the data aggregation operations. One of the distinctive features of PRIDA is that the result of the aggregation operation could be delivered to multiple DCs allowed by DOs while achieving all identified privacy goals. One can utilize a cryptographic technique alone, but it may not be able to achieve all privacy goals identified by PRIDA. When one designs a simple protocol based on **only 2PC with a single aggregator** (as in the state-of-the-art): The aggregator and DC are responsible for the aggregation operation, i.e., DC also has to participate in the aggregation computation, which is generally undesirable because DC is usually considered a party having limited resources and no expertise in privacy-preserving analytics like data owners. Preserving user privacy becomes problematic in this simple protocol. In particular, when a new DC joins the aggregation, and if a data owner (DO) does not participate in the aggregation for this DC, the aggregator easily observes that this DO does not send its data; i.e., the aggregator can **easily conduct a user profiling for data owners** whether they join or not in the aggregation operations for which DC. Even in **a 2PC-based protocol with two aggregations**, the risk of **user profiling still remains**, as DOs may decide not to send data to each new DC. Alternatively, in the case of **employing only FHE and a single aggregator**, DOs are assumed to encrypt their data with the public key of the corresponding DC, and DC can obtain the aggregated result with the help of the aggregator. In this simple solution, we re-observe that user privacy is compromised by the aggregator that can **exploit user profiling** when a data owner does not send its data to a newcomer DC. We note that one can ensure fundamental privacy properties like input privacy by using only MPC or only FHE. However, PRIDA, privacy-by-design, supports all of these privacy properties; thus, we propose the use of both techniques.

[1]https://spreadprivacy.com/how-does-google-track-me-even-when-im-not-using-it/
[2]https://www.wired.com/story/google-tracks-you-privacy/

1.2 PRIDA: Use case

When considering these newly presented privacy and security features, PRIDA for a user privacy-oriented data aggregation solution can be applied to **a variety of use cases** in which the user and its data are sensitive if compromised by the wrong parties. For example, our solution could be applied to **healthcare-related use cases**: Patients' (privacy-sensitive) data should never be shared with any third parties, such as healthcare-related centers. However, healthcare-related centers still need to be able to calculate aggregated results for new treatments, medicine stocks, experiments, statistical evaluations, etc. Note that patients may not be willing to share their data with all these healthcare-related centers, but they may want to share their data with specific centers. Patients should be able to specify which of their data should be privately shared with whom of these underlying centers. In such an example, PRIDA players are (i) a (local) hospital as Aggregator1; (ii) patients admitted to the hospital as data owners (DOs); (iii) healthcare-related centers like pharmaceutical companies, laboratories, health-oriented research centers, health insurance companies as data customers (DCs) that provide medicines, types of equipment, or insightful information about the hospital to the public. The number of patients would range from hundreds to millions [5 65]. Moreover, this local hospital may not have sufficient resources for computation capabilities or data analytics-specific expertise to process the privacy-sensitive data. Thus, a cloud provider would need to be employed, and (iv) we call it Aggregator2 in PRIDA. Patients acting as data owners (DOs) are willing to participate in a statistical evaluation over their private data: They encrypt their data and send them to the two aggregators, namely the hospital and the cloud provider. Once they receive these encrypted data, they jointly perform the required analytics over these privacy-sensitive data and send the encrypted aggregate result to the authorized centers if chosen by enough patients. Then, these authorized centers can decrypt and obtain the result. Remark that the health records of an individual are highly private, and their collection, storage, and accessibility are firmly regulated by laws on data protection [2 23 62 64]. Moreover, such details are often the target of malicious actors that attack badly protected organizations to gain an advantage through reputational damage or by requesting a ransom. In only the first 10 months of 2023, according to the Department of Health and Human Services, over 88M US individuals' medical data has been exposed [26].

PRIDA can be employed in another use case scenario, namely sports analytics since the global sports analytics market reached \$3220M in 2022 and is expected to be \$11001M by 2028 [67]. For example, multiple sports companies such as Adidas, Asics, and Nike as DCs would like to obtain some statistics (i.e., sum, average, etc.) over multiple clients' consumption or potential needs in the future, like the approaching Christmas time. They would like to learn how many pairs of shoes or which models should be delivered to their stores in a specific local. Adidas, Asics, and Nike may want to contact a sports analytics company such as Nielsen Sports (as Aggregator2, which does not have direct contact with buyers). These sports companies are competitors and may not want their competitors to know about their insights. Thus, the aggregated result should be private to each. For example, clients Alice and Bob, who are potential buyers, namely DOs for these sports companies, may like to participate in a statistical analysis with their privacy-sensitive data, such as their search (or purchase) history for running shoes or sneakers from some online marketing website like Amazon (i.e., Aggregator1 we consider and it does not have a direct contact with these sports companies). Alice usually purchases her running shoes from Amazon, while Bob uses Amazon for new sneakers. Bob always loves wearing his Adidas or Nike sneakers, whereas Alice likes running with her Asics. An analysis over the search/purchase history aggregated by two aggregators, Amazon, and Nielsen Sports, showed that Alice and Bob could need to buy new shoes because Alice's running shoes have been estimated to be used at least 500 km (since she bought them two years ago), and Bob bought his sneakers last year. As we understand it from DOs' preferences, no need for Alice is to join a survey of Adidas and Nike or no need for Bob is to join a survey of Asics. It is fair for data owners to have choices to join the analytics that may work for them. They participate in these analytics with their data (interest of buying shoes) and choice (which brand of shoes they buy) while encrypting them. Amazon and Nielsen Sports aggregate these data and could send the aggregated results to corresponding sports companies, Adidas, Asics, or Nike, which are interested in this valuable information. With the help of this information, Adidas may like to advertise its new pair of Adidas sneakers or running shoes, or Nike can publish a discount on its items for the Christmas period for their target clients, or Asics might continue selling the most sold-out model of last year this year.

We believe that thanks to these insights, data customers take action to provide better services (ads, discounts, etc.) to their target groups.

1.3 Prior Work

This section is a detailed presentation of existing privacy-preserving data aggregation solutions and highlights their relevance to PRIDA. Table 1 provides a summary of this study and regroups solutions. As shown in the table, most solutions do not focus on all identified privacy goals for PRIDA, and they result in an efficiency-oriented data aggregation enabling input privacy only.

Existing solutions rely on the use of either differential privacy (DP) mechanism [6 7 10–13 17 31–34 40 44 45 60 61 66 71 79 83], homomorphic encryption (HE) [27 39 46] or secure multi-party (or two-party) computation (MPC/2PC) [1 9 15 16 24 25 30 41 42 57 72–75 82]. Some of the DP-based solutions also implement some other cryptographic techniques such as the Paillier cryptosystem [58], etc., to ensure data confidentiality while performing data aggregation.

Table 1: Comparison of the state-of-the-art solutions.

Solutions supporting	**Prior Work**	**PRIDA**
Multiple DCs	[7 10 12 24 27 46]	✓
Input privacy	[1 6 7 9–13 15 17 24 25 27 28 30–35 39–42 44–46 52 57 61 66 68 71–75 79 82 83]	✓
Result privacy	[1 6 7 9 11 15–17 24 25 27 28 30–35 40–42 44–46 52 57 61 66 68 71–75 79 82 83]	✓
Data Control	-	✓
Anonymity of DOs	[7 10 25]	✓
Collusion-resistance	[66 68]	✓

A few works, such as [28 35 52], similar to PRIDA, combine HE and MPC to ensure data privacy. Nevertheless, these solutions do not consider the existence of multiple data customers (DCs), and they cannot be easily extended to the challenges described in Section 2. Although some prior works [7 10 12 27 46] seem to separate the role of the aggregator and DC, the result privacy is omitted. Furthermore, none of the existing solutions offers user privacy for DOs to have control over their private data while allowing data aggregation for DCs. Moreover, these solutions cannot prevent collusion between DCs and the aggregator.

Prio [24] and Prio+ [1] employ MPC: Prio uses Arithmetic shares to enable each DO to send its shares to multiple aggregators, which sum these shares locally and send back the aggregated result to DOs, and finally, DOs obtain the aggregated result. Prio presents only the computation complexity. On the other hand, Prio+, an extension of Prio, employs Boolean shares to improve the verification performance of Prio. Furthermore, some recent DP-based designs [10 13], also employing Shamir Secret Sharing (SSS) [69], involve data owners that interact with each other, which is not desired for our design: We aim to protect user privacy and enable data owners not to interact with each other or not to compute too many operations, except for just sending their privacy-sensitive data by using cryptographic techniques.

To summarize, most of the state-of-the-art do not consider multiple DCs and do not provide user privacy (data control and anonymity) for DOs, who should have the right to choose which DCs can have access to the aggregated result, including their data. Therefore, most of the existing solutions can be considered as *all-or-nothing*, whereby as soon as any DO participates in the aggregation, all DCs can have access to the aggregate result. Finally, many existing solutions do not tackle the problem of DOs' anonymity. Thus, we can conclude that by combining the use of Th-FHE and 2PC with the help of two non-colluding aggregators, PRIDA succeeds in supporting multiple DCs while ensuring user privacy (i.e., allowing DOs to control their data over DCs and to be anonymous to DCs) as well as input and result privacy.

1.4 Our contributions.

The state-of-the-art designs disregard user privacy and just focus on input privacy and enhancing the efficiency of their proposal. In this particular design, our objectives are to protect data owners, who are simply users of some application, to give back their birthright, their privacy, to them, and to minimize their workload, considering that they only encrypt and send their data. Therefore, PRIDA is designed with a user-centric approach and carefully maintains efficiency in balance with many important privacy and security goals. Our contributions can be summarized as follows:

- PRIDA helps data owners gain their **user privacy** by enabling *the control of their data*: The data owners can freely decide whether or not to participate in an aggregation operation requested by a certain data customer, and also by ensuring *the anonymity of data owners* when a data customer can have access to the aggregation result only if the number of participating DOs exceeds a predefined threshold.
- We propose to preserve **data privacy**, namely *input privacy* for DOs and *result privacy* for data customers, which generally has not been studied in the state-of-the-art solutions.
- PRIDA supports, by design, more realistic real-world use cases with **more than one data customer** thanks to the use of these two cryptographic building blocks with a setting involving two non-colluding aggregators.
- We study the existence of potential collusions among any pair of players at the very design phase, namely: Collusions between the aggregator and DOs and collusions between the aggregator and data customers. We show that PRIDA is collusion-resistant under the honest-but-curious security model.
- In order to show that PRIDA is secure and private, we conduct a detailed security analysis of PRIDA while taking potential adversaries, collusions, and itemized privacy guarantees into account.
- We implement PRIDA to evaluate its performance by conducting several experiments. The prototype implementation uses the PALISADE library [59] with the threshold version of BFV and CKKS.
- We evaluate the computational and communication costs of PRIDA, and also we compare these results with the state-of-the-art solutions [1 10 13 24]. Our performance evaluation shows that PRIDA is 17-fold better than [10 13] in timing cost.
- As previously discussed, existing solutions that do not meet the privacy goals introduced in PRIDA are not entirely suitable for a fair comparison; therefore, we compare PRIDA with two different versions based on multi-key fully homomorphic encryption, namely PRIDAv2 and PRIDAv3. Moreover, we propose a new algorithm for PRIDAv3, namely MK-TFHE . Post-process, and implement missing algorithms/gates using the prototype library of MK-TFHE. We show that PRIDA outperforms the latter two versions at least 3-fold better and 2000-fold in computational cost, respectively.

Outline. Section 2 introduces the problem of privacy-preserving data aggregation in a setting where multiple data customers are interested in the aggregation of data coming from multiple DOs. The threat model is defined in the same section. The notion of Th-FHE is defined in Section 3, and the definition of 2PC is also reminded. Section 4 describes PRIDA in detail. The security and performance evaluation of PRIDA are respectively studied in Section 5 and Section 6.

2 PROBLEM STATEMENT

A privacy-preserving data aggregation protocol is defined as a protocol where an *aggregator* collects data from a large number of clients that we name *data owners* in a privacy-preserving manner and calculates an aggregate result about these collected data without violating the individuals' privacy. The aggregation operation usually consists of the sum of the collected data. While most of the initial privacy-preserving data aggregation solutions, such as [40 66 71], let the *aggregator* have access to the aggregate information in clear, some others like [7 10 12 27 46] introduce one additional party, the *data customer*, who only receives the aggregate result (which the aggregator also has access to) and does not have any role on the aggregation operation. In this paper, instead of focusing on only an efficiency-oriented solution (which prior works proposed only), we also consider a more realistic (and more generic) scenario supporting **user privacy**, **data privacy**, and **more than one data customer** that *aggregators* may serve with the data collected from multiple data owners.

In the next section, we first identify our privacy goals raised by this setting and revise the threat model accordingly.

2.1 Privacy goals

By definition, a privacy-preserving data aggregation protocol ensures **data privacy**, namely **input privacy**, which the input data collected from multiple data owners to feed to the aggregation should remain confidential to all parties except to the actual owner. We now present **the aggregated result privacy**, which should also be confidential to the actual data customer (DC) because the introduction of **multiple data customers** may increase the risk of potential corruption among parties, which would seriously endanger the output privacy guarantees. For example, suppose a data customer and the aggregator collude; the output privacy guarantee can be in danger since the two parties could discover the result of other DC(s). This can be defined as the need for **result privacy**, which we include in the content of **data privacy**.

Furthermore, since the aggregator (Agg) is serving multiple DCs interested in receiving the aggregate result, data owners (DOs) can easily lose control on the use of their data. Indeed, DOs cannot easily **control their data** whether they have or not participated to an aggregation operation requested by a particular DC. *This problem has not been studied in the state-of-the-art solutions since those consider a unique DC or Agg who is inherently authorized to receive the aggregation result.* Moreover, in existing solutions, if DO does not want to contribute to the aggregation operation, DO simply does not send its input to Agg. This causes some problem for the solutions containing a trusted key dealer (See Section 1.3) since the aggregator does not derive the decryption key due to the failure of DO. Further, in order to fix this problem, keying materials are needed to be re-generated for the rest of DOs. Therefore, when there exist multiple DCs, the existing solutions may not be sufficient to address the control of the use of DOs' input: Authors need to update/extend their solutions. Finally, in this new setting, DOs can also wish to keep **their identity confidential to multiple DCs**: DCs should not identify which DO has participated to the actual aggregation operation. *Existing works do not discuss this problem mainly because in these solutions Agg, who obtains the aggregation* *result, usually knows all the participating DOs*. We call the two privacy goals protecting the inherent rights of data owners, namely *data control* and *anonymity*, **user privacy**.

To summarize, this new setting in privacy-preserving data aggregation introducing **multiple data customers**, on the one hand, should ensure **data privacy** by providing not only *input privacy* that is the traditional and essential privacy requirement for data owners, but also *result privacy* that is the need for data customers; on the other hand, also raises new challenges for data owners who should have **user privacy**, namely *data control of DOs over DCs* and *anonymity of DOs*.

2.2 Threat model

In this section, we define the threat model of a privacy-preserving data aggregation protocol involving multiple data customers. The introduction of multiple data customers increases the risk of potential corruption and collusion among parties, which could endanger data and user privacy guarantees.

Similar to the majority of existing studies, we assume an honest-but-curious security model where all parties have to follow the protocol steps correctly. Still, these can act curiously to infer information from the exchanged data. The potential adversaries in this threat model are enumerated as follows:

- An *external adversary* who does not participate to the protocol may try to discover information about inputs of data owners and/or results (or called outputs) of data customers.
- The honest-but-curious *Data Owner* (DO) may try to learn information about the data aggregation result(s) and other DOs' inputs. Moreover, note that DOs should not discover whether the other DOs contribute to the aggregate information or not.
- The *Aggregator* (Agg) may wish to discover input data of DOs and data customers' aggregate results.
- *Data Customer* (DC) may try to discover information about inputs of DOs and other DCs' aggregate results. DC may also try to discover whether DOs contribute to the aggregate information or not.

Also, DC_j may wish to learn the aggregate result when the number of DOs who choose that DC is not at least as some threshold t. When data customer DC_j can reach or be above threshold t, this makes DC_j *"authorized"* to receive the aggregation result. In other words, DO does not have to send its sensitive data for each DC and can choose which DC could have the aggregated result, consisting of its private data.

PRIDA: Overview.

In order to cope with the challenges identified in the previous section, namely (i) ***user privacy***, i.e. the lack of data control of DOs and the anonymity of DOs; (ii) data privacy, namely ***result privacy*** in addition to input privacy; and (iii) the possible collusions between parties, we propose a new solution named PRIDA that firstly introduces two non-colluding aggregators instead of one aggregator. This new setting protects against collusions among different parties. Furthermore, to ensure both input and result privacy and enable DO to control its data over DCs and be anonymous to DCs, we propose to combine the use of threshold fully homomorphic encryption (Th-FHE) whereby one can perform operations over data encrypted under multiple keys, and secure two-party computation where two parties jointly perform some function over their private inputs without disclosing them.

Furthermore, in order to protect the anonymity of DOs, before the actual privacy-preserving aggregation operation, we define a preliminary privacy-preserving counting scheme that ensures that aggregation takes place only if some threshold number is reached.

PRIDA involves the following three parties in Figure 1:

- *Data Owner* (DO) owns some confidential input data and outsources this confidential data to the aggregators once its input data is first secretly shared and further encrypted with Th-FHE. DO also defines which data customer can access the aggregate result involving its input by employing another private data named choice data.
- The two *Aggregators* (Agg1 and Agg2) are non-colluding cloud servers that collect the encrypted data from multiple DOs, perform data aggregation, and further, one (Agg1) helps DOs collect their data, and the other one (Agg2) helps data customers receive the result.
- *Data Customer* (DC) obtains the data aggregation result over the inputs from multiple DOs in cleartext if authorized, using the decryption algorithm of Th-FHE.

Colluding adversaries in PRIDA.

As previously mentioned, the existence of multiple DCs may increase the power of adversaries through potential collusions. For example, if some DC and Agg2 collude, they can try to discover the input of some DOs. Also, the result privacy guarantee can be in danger since these two parties can disclose the result of other DC(s) when they collude. Therefore, the threat model should also consider the collusions between parties:

- When *the curious DO and Agg1 collude*, they should not learn any information about other DOs' inputs and DCs' data aggregation results.
- If *DC and Agg2 collude*, they should not learn anything about inputs of DOs and other DCs' aggregation results.
- Finally, Agg1 and Agg2 should not discover the input of DOs and the result of DCs' even if *DC colludes with Agg2* and/or *DO colludes with Agg1*.

Remarks on other collusions.

If **DO and Agg2 collude**, they can try to learn something about inputs of other DOs or aggregation results of DCs; yet, they cannot discover anything about them since inputs of DOs, namely the input data and the choice data, are shared-encrypted or secretly shared and also results of DCs are encrypted. More importantly they do need to have all decryption keying material to decrypt those data and the other shared data. The only information Agg2 obtains is the identity, and inputs of the colluding DO if it lets. There is in this case no need for privacy protection of this DO which has already been in collaborating with Agg2. Note that, in the collusion of DO-Agg2, the compromised DO cannot collude with Agg1 (and Agg2 not collude with DC) at the same time.

When **DC and Agg1 collude**, they can try to obtain some information about inputs of DOs or other DCs' aggregation results. Similar to the DO-Agg2 collusions, they cannot discover anything about them since they neither do have all decryption keying material nor the complementary of shared data. The collusions of

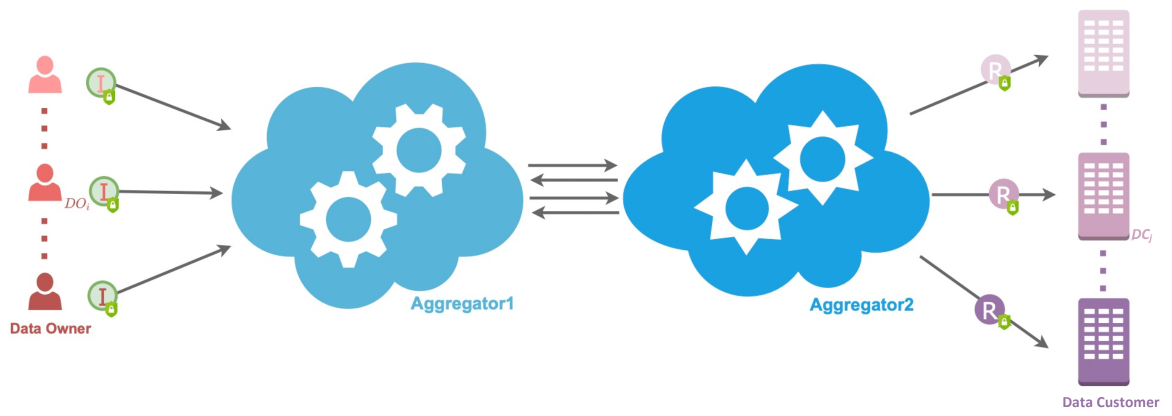


Figure 1: PRIDA players.

DC-Agg1 can put the anonymity of DOs, which rely on Agg1, in danger, which contradicts the threat model of PRIDA. We have aimed at being a privacy advocate for data owners who deserve their data control and anonymity as well as data privacy in PRIDA. Therefore, we consider *collusions of DC-Agg1 should not happen.*

3 PRELIMINARIES

This section introduces the building blocks of PRIDA, namely: Secure two-party computation and threshold fully homomorphic encryption.

Notation. We denote vectors within bold: $\mathbf{v}$. We utilize $\star$ for the star product of two vectors which is the element-wise multiplication of two vectors, and $\cdot$ for a simple multiplication of two elements of the vectors. $[.]$ denotes the Th-FHE-encrypted data. Index i in $\{1, \ldots, n\}$ represents data owner DO_i whereas index $j \in \{1, \ldots, m\}$ represents data customer DC_j with $n, m \in \mathbb{Z}^+$ denoting the total number of DOs and DCs, respectively.

3.1 Secure Two-party Computation

Secure two-party computation (2PC) is a technique in which two parties can jointly perform a function over private inputs without revealing any information but the result. In our work, PRIDA uses arithmetic secret sharing.

Arithmetic secret sharing. Arithmetic secret sharing allows two parties to compute additions and multiplications over secretly shared values. We denote $\langle . \rangle_k$, $k = 1, .., p$, to represent an arithmetic secret share. A secret input d is split into p shares, using $\text{AS.Share}(p, d)$ which returns $\langle d \rangle_1, \ldots, \langle d \rangle_p$ such that $\langle d \rangle_1 + \ldots + \langle d \rangle_p \equiv d \bmod 2^l$ where l is the bit size for 2PC operations. The addition over the shared values is computed locally whereas performing the multiplication operation requires two parties to interact. This can be performed with Beaver's multiplication triplets [8].

3.2 Threshold Fully Homomorphic Encryption

Due to the existence of multiple DCs and the two aggregators, we propose to employ a particular homomorphic encryption scheme, namely threshold fully homomorphic encryption [4 49].

Threshold fully homomorphic encryption (Th-FHE) is a homomorphic encryption scheme whereby multiple parties jointly generate a common (unique) public key using their individual public and secret keys. The resulting secret key is not disclosed to any party. The party encrypts its private data using the common public key, and operations are performed over encrypted data. In order to decrypt the result, these parties contribute to the decryption of the output: First, they partially decrypt with their secret keys and merge the partially decrypted values.

A semantically secure Th-FHE scheme contains six probabilistic polynomial-time algorithms:

- $pp \leftarrow \text{Th-FHE.Setup}(1^\lambda)$: This algorithm outputs public parameters pp given security parameter $\lambda \in \Lambda$.
- $(pk, sk_{id_1}, \ldots, sk_{id_k}) \leftarrow \text{Th-FHE.KeyGen}(pp)$: This randomized algorithm generates k secret keys sk_{id_i} and a common public key pk given the public parameters pp.
- $\text{ct} \leftarrow \text{Th-FHE.Encrypt}(\text{m}, pk)$: This randomized algorithm encrypts message m with public key pk and outputs ciphertext ct.
- $\text{ct}^* \leftarrow \text{Th-FHE.Eval}(C, (\text{ct}_1, \ldots, \text{ct}_l))$: This algorithm evaluates circuit C over the l ciphertexts.
- $\mu_i \leftarrow \text{Th-FHE.PartialDecrypt}(\text{ct}^*, sk_{id_i})$: This algorithm takes as input ciphertext ct^* and secret key sk_{id_i} of party i, and outputs a partially decrypted information μ_i.
- $\text{m}' \leftarrow \text{Th-FHE.Merge}(\mu_1, \ldots, \mu_k)$: This algorithm takes as input all the partial decryptions derived from a ciphertext ct^* and outputs the final plaintext m'.

For more details on Th-FHE, we refer readers to [63].

4 PRIDA: PRIVACY-PRESERVING DATA AGGREGATION

We present PRIDA, which combines the use of 2PC with threshold fully homomorphic encryption (Th-FHE) to ensure that (i) ***user privacy***: Data owners can choose which data customer can have access to the aggregate result involving their input **without being profiled** and **with being anonymized to multiple data customers** while joining in the data aggregation; (ii) ***data privacy***:

Data owners can keep their sensitive data private and also an authorized data customer can receive the aggregation result private only to itself; and (iii) ***multiple data customers*** can be supported. As previously mentioned, the only assumption we make is that the two aggregators do not collude.

PRIDA is constructed using 2PC and Th-FHE: PRIDA employs Th-FHE differently than proposed [4 49] since the one who encrypts its data does not use its public key. Instead, privacy-sensitive data are encrypted with one unique public key collaboratively generated by each DC and the two aggregators. That is, we propose PRIDA, which consists of each data owner (DO) encrypting the data with this common public key. The decryption involves the three secret keys of these three parties corresponding to this public key. Thus, in order to protect input data privacy, before encryption, we newly propose that DO splits its data into two shares, encrypts each data with this public key, and further sends the encrypted data to the aggregators.

PRIDA is defined in five phases, as also presented in Protocol 1:

1. *The setup phase* in which the keying material is generated.
2. *The data protection phase*, whereby DO actually decides which DC can have access to the aggregate result of which DO has participated, and encrypts and secretly shares the relevant input data accordingly.
3. *The preliminary counting phase*, whereby the two aggregators find out which DC can receive the aggregated result by counting the number of DOs and verifying if this number exceeds a predefined threshold.
4. *The aggregation phase*, which consists of the two aggregators jointly aggregating the data collected from different DOs.
5. *The decryption phase*, in which an authorized DC decrypts the data aggregation result with the help of the two aggregators.

Algorithm 1: Aggregation phase

```
1  Inputs. $Agg_k$ ($k \in \{1,2\}$) has $[\langle \mathbf{dv}_i \rangle_k]$, $\langle \mathbf{cv}_i \rangle_k$, $\langle \boldsymbol{\alpha}_i \rangle_k$,
     $\langle \boldsymbol{\beta}_i \rangle_k$, $\langle \boldsymbol{\gamma}_i \rangle_k$.
2  Output. $Agg_k$ calculates the result $[s_j]$ for authorized $DC_j$.
3  Algorithm steps. for $i = 1$ to $n$ do
4      $Agg_k$: $[\langle \boldsymbol{\epsilon}_i \rangle_k] \leftarrow$ Eval(+, ($[\langle \mathbf{dv}_i \rangle_k]$,$\langle \boldsymbol{\alpha}_i \rangle_k$)).
5      $Agg_k$: $\langle \boldsymbol{\delta}_i \rangle_k \leftarrow \langle \mathbf{cv}_i \rangle_k + \langle \boldsymbol{\beta}_i \rangle_k$.
6      $Agg_k$: Exchange $[\langle \boldsymbol{\epsilon}_i \rangle_k]$ and $\langle \boldsymbol{\delta}_i \rangle_k$ to compute $[\boldsymbol{\epsilon}_i]$ and
         $\boldsymbol{\delta}_i$, where $k = 1, 2$.
7      $Agg_k$: $[\boldsymbol{\epsilon}_i \star \langle \mathbf{cv}_i \rangle_k] \leftarrow$ Eval($\star$, ($[\boldsymbol{\epsilon}_i]$, $\langle \mathbf{cv}_i \rangle_k$)).
8      $Agg_k$: $[\boldsymbol{\delta}_i \star \langle \mathbf{dv}_i \rangle_k] \leftarrow$ Eval ($\star$, ($[\langle \mathbf{dv}_i \rangle_k]$, $\boldsymbol{\delta}_i$)).
9      $Agg_k$: $[\langle \boldsymbol{\gamma}_i \rangle_k + \boldsymbol{\epsilon}_i \star \langle \mathbf{cv}_i \rangle_k + \boldsymbol{\delta}_i \star \langle \mathbf{dv}_i \rangle_k] \leftarrow$
         Eval(+, ($[\boldsymbol{\epsilon}_i \star \langle \mathbf{cv}_i \rangle_k]$, $[\boldsymbol{\delta}_i \star \langle \mathbf{dv}_i \rangle_k]$, $\langle \boldsymbol{\gamma}_i \rangle_k$)).
10     Agg1: Send $[\langle \boldsymbol{\gamma}_i \rangle_1 + \boldsymbol{\epsilon}_i \star \langle \mathbf{cv}_i \rangle_1 + \boldsymbol{\delta}_i \star \langle \mathbf{dv}_i \rangle_1]$ to Agg2.
11     Agg2: $[\boldsymbol{\gamma}_i + \boldsymbol{\epsilon}_i \star \mathbf{cv}_i + \boldsymbol{\delta}_i \star \mathbf{dv}_i] \leftarrow$ Eval(+, ($[\langle \boldsymbol{\gamma}_i \rangle_1 + \boldsymbol{\epsilon}_i \star$
         $\langle \mathbf{cv}_i \rangle_1 + \boldsymbol{\delta}_i \star \langle \mathbf{dv}_i \rangle_1]$, $[\langle \boldsymbol{\gamma}_i \rangle_2 + \boldsymbol{\epsilon}_i \star \langle \mathbf{cv}_i \rangle_2 + \boldsymbol{\delta}_i \star \langle \mathbf{dv}_i \rangle_2]$)).
12     Agg2: $[\boldsymbol{\epsilon}_i \star \boldsymbol{\delta}_i] \leftarrow$ Eval($\star$, ($[\boldsymbol{\epsilon}_i]$, $\boldsymbol{\delta}_i$)).
13     Agg2:
         $[\mathbf{dv}_i \star \mathbf{cv}_i] \leftarrow$ Eval(−, ($[\boldsymbol{\gamma}_i + \boldsymbol{\epsilon}_i \star \mathbf{cv}_i + \boldsymbol{\delta}_i \star \mathbf{dv}_i]$, $[\boldsymbol{\epsilon}_i \star \boldsymbol{\delta}_i]$)).
14 Agg2: Call Eval for adding all $[d_{ij} \cdot c_{ij}]$ to find $[s_j]$.
```

Protocol 1 PRIDA

Inputs. DO_i, $i \in \{1, \ldots, n\}$, inputs a choice vector $\mathbf{cv}_i$ and a data vector $\mathbf{dv}_i$ of size m. Also, a pre-defined threshold t and the public parameters pp are published.

Output. If $cv_{total_j} \geq t$, DC_j obtains the aggregate result s_j, $j \in \{1, \ldots, m\}$. Otherwise, DC_j obtains nothing.

Protocol steps.

1. *Setup executed by DC_j, Agg1, and Agg2.*
   a. $(pk, sk_{DC_j}, sk_{Agg1}, sk_{Agg2}) \leftarrow$ Th-FHE . KeyGen(pp).
2. *Data protection executed by DO_i.*
   a. $(\langle \mathbf{cv}_i \rangle_1, \langle \mathbf{cv}_i \rangle_2) \leftarrow$ AS. Share(2, $\mathbf{cv}_i$).
   b. $(\langle \mathbf{dv}_i \rangle_1, \langle \mathbf{dv}_i \rangle_2) \leftarrow$ AS. Share(2, $\mathbf{dv}_i$).
   c. $[\langle dv_{ij} \rangle_k] \leftarrow$ Th-FHE . Encrypt($\langle dv_{ij} \rangle_k$, pk), $k = 1, 2$.
   d. Generate random vectors $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$ and compute $\boldsymbol{\gamma}_i$ such that $\boldsymbol{\gamma}_i = \boldsymbol{\alpha}_i \star \boldsymbol{\beta}_i$.
   e. *Beaver's triplets.* Call AS. Share(2, .) for $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, $\boldsymbol{\gamma}_i$.
   f. Send $[\langle \mathbf{dv}_i \rangle_k]$, $\langle \mathbf{cv}_i \rangle_k$, $\langle \boldsymbol{\alpha}_i \rangle_k$, $\langle \boldsymbol{\beta}_i \rangle_k$, and $\langle \boldsymbol{\gamma}_i \rangle_k$ to Agg$k$, $k = 1, 2$.
3. *Preliminary counting executed by Agg1 and Agg2.*
   a. Agg$k$: Obtain $\langle \mathbf{cv}_{total} \rangle_k = \sum \langle \mathbf{cv}_i \rangle_k$, $k = 1, 2$.
   b. Agg$k$: Exchange $\langle \mathbf{cv}_{total} \rangle_k$ to get $\mathbf{cv}_{total} = (cv_{total_1}, \ldots, cv_{total_j}, \ldots, cv_{total_m})$.
   c. Agg$k$: Label DC_j as authorized if $cv_{total_j} \geq t$.
4. *Aggregation executed by Agg1 and Agg2.*
   a. Jointly compute $[\mathbf{s}] = [\sum \mathbf{dv}_i \star \mathbf{cv}_i]$ for each authorized DC_j. The details of data aggregation are provided in Algorithm 1.
5. *Decryption executed by Agg1, Agg2, and DC_j.*
   a. Agg2: Send the aggregate result vector $[\mathbf{s}] = (\ldots, [s_j], \ldots)$ to Agg1.
   b. Agg1, Agg2: Employ $\mu_k \leftarrow$ Th-FHE . PartialDecrypt to decrypt $[\mathbf{s}]$ where $k = 1, 2$.
   c. Agg1: Send μ_1 to Agg2.
   d. Agg2: Send μ_1 and μ_2 to the authorized DC_j.
   e. DC_j: Call $\mu_3 \leftarrow$ Th-FHE . PartialDecrypt.
   f. DC_j: Run Th-FHE . Merge with μ_1, μ_2, μ_3 to find s_j.

In more details, during the *setup phase*, each DC and two aggregators jointly generate the unique keying material according to threshold homomorphic encryption. In the second phase, **each DO_i decides which DC is authorized to access the aggregate result in which its input is involved**. With this aim, DO_i first defines a binary **choice vector $\mathbf{cv}_i$**= $(cv_{i1}, ..., cv_{ij}, ..., cv_{im})$, where each element is mapped to a particular DC_j, and the value corresponds to the authorization decision ($cv_{ij} = 0$ if DC_j is not authorized, and 1 if authorized). Each DO also defines a *data vector* **dv** (same size as **cv**) for its input data. If the actual DC_j is authorized, then the corresponding element dv_{ij} is set to the private input of DO_i. Otherwise, DO generates a random number r, which is set to the corresponding element of vector **dv**. Finally, DO randomly generates two arithmetic secret shares for **cv** and **dv** and further encrypts the two shares of **dv**. Each share is then sent to the corresponding aggregator (Agg$k$, k = 1 or 2). Then, in the *preliminary counting phase*, Agg1 and Agg2 jointly add the choice vectors received from each DO in 2PC and further obtain the resulting number of DOs per DC_j, $\mathrm{cv}_{total_j} = \sum \mathrm{cv}_{ij}$, without discovering which DO has authorized which DC. If the number of DOs authorizing a particular DC_j is greater than the pre-defined threshold t, then the two aggregators can start the actual *aggregation phase* defined in Algorithm 1: Aggregators simply compute $\mathrm{cv}_{ij} \cdot [\mathrm{dv}_{ij}]$ for each DO_i in 2PC and further compute the sum of these intermediate values for all DO_i to find aggregate result $[s_j] = [\sum \mathrm{dv}_{ij} \cdot \mathrm{cv}_{ij}]$ for authorized DC_j.

Finally, during the *decryption phase*, the aggregators partially decrypt the aggregate result, which is further sent to the corresponding authorized DC, which is now its turn to partially decrypt and obtain the plaintext result s_j.

To fully achieve the privacy goals established for PRIDA, it is insufficient to rely solely on 2PC or Th-FHE. This is due to the fact that if 2PC is implemented alone, unauthorized DCs could obtain the aggregate result by collaborating with a single Agg, while Th-FHE alone could expose the inputs of DOs and render it challenging for DOs to control their data in the problem of collusion between Agg and DC.

5 SECURITY ANALYSIS

We analyze the security of PRIDA, taking the previously introduced threat model into account, and show that PRIDA satisfies the privacy goals defined in Section 2. There exists a strong relationship between user and data privacy requirements, such that they are deeply intertwined. This implies that cryptographic primitives utilized to ensure user privacy also play a crucial role in safeguarding data privacy and vice-versa. We also study the analysis by considering each player as an attacker in Appendix A. PRIDA aims to compute privacy-preserving data aggregation in the honest-but-curious adversarial model, and we assume the honest-but-curious adversary is non-adaptive and computationally bounded. The definitions A.1, A.2, and A.3 in Appendix A help us prove that PRIDA is secure and private. We guarantee secure and private aggregation thanks to the cryptographic techniques we employ to design PRIDA. Both threshold homomorphic encryption (Th-FHE) and secure two-party computation (2PC) are proven to be secure. Indeed, the Th-FHE scheme is semantically secure [4 49].

In the data protection and decryption phases of PRIDA, the security is achieved by the semantic security (in Definition A.3) of the Th-FHE scheme and/or the information-theoretic security of 2PC since data and choice vectors are firstly secretly shared, and the data vectors are encrypted. Note that the Th-FHE scheme satisfies semantic security against chosen plaintext attacks under the (ring) learning with errors. Hence, these shared and/or shared-encrypted data are indistinguishable for all parties except the actual DO. Those parties obtain only indistinguishable random values that do not disclose any information about the input data and the aggregation result. When we investigate the security of encryption and decryption of the input data and the result of private aggregation that meets Definition A.3, in the phases of data protection and decryption, we now provide the simulation for the data protection phase only in this section, and the decryption phase is the reverse operation of the encryption.

Simulator $\mathcal{A}'$**.** With holding inputs pp and $1^{|X_\lambda|}$, where λ is the security parameter, Algorithm $\mathcal{A}'$ performs the following:

1. $pk \leftarrow$ Th-FHE. KeyGen(pp) to generate a unique public key pk with the given information pp and λ.
2. $[0^{|X_\lambda|}]_{pk} \leftarrow$ Th-FHE. Encrypt$(0^{|X_\lambda|}, pk)$ to obtain the encryption of "garbage" based on the given information $1^{|X_\lambda|}$.
3. Lastly, $\mathcal{A}'$ runs the algorithm $\mathcal{A}(1^\lambda, \mathrm{ct}_\lambda, 1^{|X_\lambda|})$, where $\mathrm{ct}_\lambda = [0^{|X_\lambda|}]_{pk}$, and receives whatever $\mathcal{A}$ obtains.

When $\mathcal{A}$ runs the encryption algorithm Th-FHE. Encrypt on input X_λ (i.e., $\mathcal{A}$ obtains $[X_\lambda]_{pk} \leftarrow$ Th-FHE. Encrypt(X_λ, pk)), $\mathcal{A}'$ gets output ct_λ. That is, the simulation has performed the encryption over input zeros. If the encryption scheme is *indistinguishable*, then $\mathcal{A}$ obtains $f(1^\lambda, X_\lambda)$ (which equals to $(1^\lambda, [X_\lambda]_{pk}, 1^{|X_\lambda|})$) with almost the same probability when given Th-FHE. Encrypt(X_λ, pk) like the given information Th-FHE. Encrypt$(0^{|X_\lambda|}, pk)$. The simulator simulates an execution for the adversary using the indistinguishable encryption of zeros. Therefore, $\mathcal{A}$ cannot distinguish between these two encryptions.

The preliminary counting phase, on the other hand, requires the two aggregators to jointly compute addition over DOs' choice vectors, and the security of the addition operation is guaranteed using secure two-party computation. Moreover, the two aggregators perform the data aggregation phase using arithmetic secret sharing via Beaver triplets [8] over Th-FHE encrypted data. In addition to the security of Th-FHE, arithmetic secret sharing used in the preliminary counting and data aggregation phases of PRIDA as a secure two-party computation technique achieve indistinguishability given that the shares are generated from a uniformly random distribution [37]. Similarly, those data do not leak anything about inputs and aggregation results; only the authorized (i.e., actual) DC discovers the aggregate result. In order to prove the formal security to show that PRIDA performs private data aggregation securely, we use the simulation proof technique [48].

The simulation proof technique is a method to analyze security in two worlds, namely the ideal and real worlds. A protocol is considered secure when the adversaries in the two worlds learn the same amount of knowledge, i.e., approximately nothing during the protocol runs. The ideal world security is established by outsourcing inputs of two parties to a trusted third party that can perform the computations and return the result, whereas the security of the real world is provided if an adversary $\mathcal{A}$ can attack the protocol in the real world, then the attack can also be performed by an adversary $\mathcal{S}$ in the ideal world. Since the attacks of $\mathcal{S}$ are not successful in the ideal world, the attacks in the real world also fail, and the protocol is proved to be secure in the real-world setting as in Definition A.4.

Accordingly, Algorithm 1 is a protocol π between Agg1 and Agg2, which computes the functionality f that aggregates private inputs of DOs. In more details, Agg1 provides shared and shared-encrypted inputs, namely $x_{i1} = \{\langle \mathbf{cv}_i \rangle_1, [\langle \mathbf{dv}_i \rangle_1], \langle \boldsymbol{\alpha}_i \rangle_1, \langle \boldsymbol{\beta}_i \rangle_1, \langle \boldsymbol{\gamma}_i \rangle_1\}$ for π, and Agg2's inputs are $x_{i2} = \{\langle \mathbf{cv}_i \rangle_2, [\langle \mathbf{dv}_i \rangle_2], \langle \boldsymbol{\alpha}_i \rangle_2, \langle \boldsymbol{\beta}_i \rangle_2, \langle \boldsymbol{\gamma}_i \rangle_2\}$, where $\mathbf{cv}_i$, $\mathbf{dv}_i$, $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, and $\boldsymbol{\gamma}_i$ are split into two shares, and further, shared $\mathbf{dv}_i$ is encrypted under Th-FHE. Given x_{i1}, x_{i2}, f computes $f(x_{i1}, x_{i2}) = (\perp, [s_j])$, where $[s_j] = \sum [d_{ij} \cdot c_{ij}]$, and Agg1 receives nothing at the end of data aggregation, i.e., its output is an empty string $\perp$, and Agg2 receives the encrypted aggregation result for the actual DC.

Theorem 5.1. *The aggregation protocol π (Algorithm 1) securely computes the functionality $f(x_{i1}, x_{i2}) = (\bot, [s_j])$ in the presence of honest-but-curious, non-adaptive, and computationally bounded adversaries.*

Proof. The proof and demonstration are provided by the ideal-real simulation, which illustrates that adversaries cannot differentiate between "real" world experiments, where they receive "real" data, and "ideal" world experiments, where they receive random data generated by the simulator(s). Theorem 5.1 is proved below for corrupted Agg1 and Agg2, by demonstrating that Adversary $\mathcal{A}$'s views under the real-world conditions are computationally indistinguishable from the simulated views of $\mathcal{S}_i$, where $i \in \{1, 2\}$ is for the Agg1 and Agg2, respectively. We construct a separate simulator for each party.

- **Agg1 is corrupted by $\mathcal{A}$**: It is important that Agg1 receives no output. So, we need to show that Simulator $\mathcal{S}_1$ can generate the view of incoming messages to Agg1, which are $[\langle\epsilon_i\rangle_2]$ and $\langle\delta_i\rangle_2$. Then, we can conclude by comparing the real view and the simulated view. $\mathcal{S}_1$ is given with the security parameter λ and inputs, namely $x_{i1} = \{\langle\mathbf{cv}_i\rangle_1, [\langle\mathbf{dv}_i\rangle_1], \langle\alpha_i\rangle_1, \langle\beta_i\rangle_1, \langle\gamma_i\rangle_1\}$ and works as follows:
  1. $\mathcal{S}_1$ chooses a uniformly distributed random tape r'_1.
  2. $\mathcal{S}_1$ guesses intermediate messages $[\langle\epsilon'_i\rangle_2]$ and $\langle\delta'_i\rangle_2$ using random tape r'_1.
  3. $\mathcal{S}_1$ outputs $[\langle\epsilon'_i\rangle_2]$ and $\langle\delta'_i\rangle_2$.

  The view of Agg1, $\mathbf{view}_1^{\pi}(x_{i1}, x_{i2})$, in the real world is
  $$\left\{\langle\mathbf{cv}_i\rangle_1, [\langle\mathbf{dv}_i\rangle_1], \langle\alpha_i\rangle_1, \langle\beta_i\rangle_1, \langle\gamma_i\rangle_1, r_1; [\langle\epsilon_i\rangle_2], \langle\delta_i\rangle_2\right\}, \quad (1)$$
  while the view generated by the simulator, $\mathcal{S}_1(1^\lambda, x_{i1}, f_1(x_{i1}, x_{i2}))$,
  $$\left\{\langle\mathbf{cv}_i\rangle_1, [\langle\mathbf{dv}_i\rangle_1], \langle\alpha_i\rangle_1, \langle\beta_i\rangle_1, \langle\gamma_i\rangle_1, r'_1; [\langle\epsilon'_i\rangle_2], \langle\delta'_i\rangle_2\right\}. \quad (2)$$
  Thanks to the security of AS. Share and Th-FHE. Decrypt, simulator, $\mathcal{S}_1$ cannot distinguish the real $\{[\langle\epsilon_i\rangle_2], \langle\delta_i\rangle_2\}$ from uniformly random $\{[\langle\epsilon'_i\rangle_2], \langle\delta'_i\rangle_2\}$. So we obtain
  $$\left\{\mathcal{S}_1(1^\lambda, x_{i1}, f_1(x_{i1}, x_{i2}))\right\} \overset{c}{\equiv} \left\{\mathbf{view}_1^{\pi}(x_{i1}, x_{i2})\right\}. \quad (3)$$
- **Agg2 is corrupted by $\mathcal{A}$**: Simulator $\mathcal{S}_2$ requires the construction of a view for the real and ideal worlds. Observe that a party's view includes its input, random tape, and all intermediate messages. The result obtained by running the protocol instructions on the simulator view has to be correct; otherwise, the distinguisher can easily determine that the result does not correspond to the view of the real-world execution. $\mathcal{S}_2$ receives the security parameter λ, Agg2's input $x_{i2} = \{\langle\mathbf{cv}_i\rangle_2, [\langle\mathbf{dv}_i\rangle_2], \langle\alpha_i\rangle_2, \langle\beta_i\rangle_2, \langle\gamma_i\rangle_2\}$ and the aggregation result $[s] = \sum[\mathbf{dv}_i \star \mathbf{cv}_i]$ and thus, $\mathcal{S}_2$ works as follows:
  1. $\mathcal{S}_2$ chooses a uniformly distributed random tape r'_2.
  2. $\mathcal{S}_2$ guesses intermediate messages $[\langle\epsilon'_i\rangle_1]$ and $\langle\delta'_i\rangle_1$ using random tape r'_2.
  3. $\mathcal{S}_2$ computes $[\langle\epsilon_i\rangle_2] = \text{Eval}(+, ([\langle\mathbf{dv}_i\rangle_2], \langle\alpha_i\rangle_2))$ and $\langle\delta_i\rangle_2 = \langle\mathbf{cv}_i\rangle_2 + \langle\beta_i\rangle_2$.
  4. $\mathcal{S}_2$ obtains $[\epsilon'_i] = \text{Eval}(+, ([\langle\epsilon'_i\rangle_1], [\langle\epsilon_i\rangle_2]))$ and $\delta'_i = \langle\delta'_i\rangle_1 + \langle\delta_i\rangle_2$.
  5. $\mathcal{S}_2$ estimates $[(\epsilon_i \star \delta_i)'] = \text{Eval}(\star, ([\epsilon'_i], \delta'_i))$ like the real Agg2.
  6. $\mathcal{S}_2$ computes $[\rho'_2] = [\langle\gamma_i\rangle_2 + \epsilon'_i \star \langle\mathbf{cv}_i\rangle_2 + \delta'_i \star \langle\mathbf{dv}_i\rangle_2]$.
  7. $\mathcal{S}_2$ sets $[\rho'] = [(\gamma_i + \epsilon_i \star \mathbf{cv}_i + \delta_i \star \mathbf{dv}_i)']$ $= \text{Eval}(-, ([s], [(\epsilon_i \star \delta_i)']))$, where $[s] = [\sum \mathbf{dv}_i \star \mathbf{cv}_i]$ is received from Agg2.
  8. $\mathcal{S}_2$ calculates $[\rho'_1] = [(\langle\gamma_i\rangle_1 + \epsilon_i \star \langle\mathbf{cv}_i\rangle_1 + \delta_i \star \langle\mathbf{dv}_i\rangle_1)']$ $= Eval(-, ([\rho'], [\rho'_2]))$.
  9. $\mathcal{S}_2$ outputs $[\langle\epsilon'_i\rangle_1]$, $\langle\delta'_i\rangle_1$ and $[\rho'_1]$.

  The view of Agg2, $\mathbf{view}_2^{\pi}(x_{i1}, x_{i2})$, in the real world is
  $$\left\{\langle\mathbf{cv}_i\rangle_2, [\langle\mathbf{dv}_i\rangle_2], \langle\alpha_i\rangle_2, \langle\beta_i\rangle_2, \langle\gamma_i\rangle_2, r_2; [\langle\epsilon_i\rangle_1], \langle\delta_i\rangle_1, \rho_1\right\}, \quad (4)$$
  while the view generated by the simulator, $\mathcal{S}_2(1^\lambda, x_{i2}, f_2(x_{i1}, x_{i2}))$,
  $$\left\{\langle\mathbf{cv}_i\rangle_2, [\langle\mathbf{dv}_i\rangle_2], \langle\alpha_i\rangle_2, \langle\beta_i\rangle_2, \langle\gamma_i\rangle_2, r'_2; [\langle\epsilon'_i\rangle_1], \langle\delta'_i\rangle_1, \rho'_1\right\}. \quad (5)$$
  Simulator $\mathcal{S}_2$ cannot accurately predict $[\langle\epsilon'_i\rangle_1]$, $\langle\delta'_i\rangle_1$, $\langle\gamma'_i\rangle$ from ρ'_1, and therefore $[\langle\epsilon'_i\rangle]$ and $\langle\delta'_i\rangle$ as well. Indeed, $\mathcal{S}_2$ cannot calculate any intermediate value based on $[\langle\epsilon'_i\rangle_1]$, and $\langle\delta'_i\rangle_1$ as the way Agg2 does. However, we can say that the distribution of $[\langle\epsilon'_i\rangle_1]$ and $\langle\delta'_i\rangle_1$ will be very close to the distribution of $[\langle\epsilon_i\rangle_1]$ and $\langle\delta_i\rangle_1$ because of the random uniformly distributed coefficients of the Beaver's triplet. Moreover, like simulator $\mathcal{S}_1$, $\mathcal{S}_2$ cannot distinguish the real and ideal case thanks to the security of AS. Share and Th-FHE. Decrypt. Therefore, we obtain
  $$\left\{\mathcal{S}_2(1^\lambda, x_{i2}, f_2, f(x_{i1}, x_{i2}))\right\} \overset{c}{\equiv} \left\{\mathbf{view}_2^{\pi}(x_{i1}, x_{i2}), \mathbf{output}^{\pi}(x_{i1}, x_{i2}, \lambda)\right\}, \quad (6)$$
  We can now estimate the advantage of an attacker in distinguishing views under the real-world setting in a way that there exists a non-uniform polynomial-time distinguisher D with a negligible function $\mu(\lambda)$ such that
  $$\Big|\Pr\left[D\left(\langle\mathbf{cv}_i\rangle_2, [\langle\mathbf{dv}_i\rangle_2], \langle\alpha_i\rangle_2, \langle\beta_i\rangle_2, \langle\gamma_i\rangle_2, r'_2; [\langle\epsilon'_i\rangle_1], \langle\delta'_i\rangle_1, \rho'_1\right) = 1\right] - \Pr\left[D\left(\langle\mathbf{cv}_i\rangle_2, [\langle\mathbf{dv}_i\rangle_2], \langle\alpha_i\rangle_2, \langle\beta_i\rangle_2, \langle\gamma_i\rangle_2, r_2; [\langle\epsilon_i\rangle_1], \langle\delta_i\rangle_1, \rho_1\right) = 1\right]\Big| \leq \mu(\lambda). \quad (7)$$
  In Equation 7, the negligible function $\mu(\lambda)$, which is an extremely small value, indicates that the simulator exists and generates a view for the adversary in the real world that is computationally indistinguishable from its real version.
  As a result of the computational indistinguishability in Definition A.1, both views of $\mathcal{S}_2$ and Agg2 are naturally nonuniform. We have shown that views of $\mathcal{S}_2$ and Agg2 are only computationally indistinguishable by employing the negligible function $\mu(\lambda)$ in order to demonstrate their non-uniformity. When we rewrite indistinguishability for every non-uniform polynomial-time algorithm D and every polynomial $p(\lambda)$, there exists λ such that,
  $$\Big|\Pr\left[D\left(\langle\mathbf{cv}_i\rangle_2, [\langle\mathbf{dv}_i\rangle_2], \langle\alpha_i\rangle_2, \langle\beta_i\rangle_2, \langle\gamma_i\rangle_2, r'_2; [\langle\epsilon'_i\rangle_1], \langle\delta'_i\rangle_1, \rho'_1\right) = 1\right] - \Pr\left[D\left(\langle\mathbf{cv}_i\rangle_2, [\langle\mathbf{dv}_i\rangle_2], \langle\alpha_i\rangle_2, \langle\beta_i\rangle_2, \langle\gamma_i\rangle_2, r_2; [\langle\epsilon_i\rangle_1], \langle\delta_i\rangle_1, \rho_1\right) = 1\right]\Big| < \frac{1}{p(\lambda)}. \quad (8)$$
  Equations 7 and 8 guarantee that a corrupted Agg2 has no advantage on distinguishability.

□

Protocol π between Agg1 and Agg2 is secure; further, we can conclude that PRIDA provides secure data aggregation, including input privacy and aggregation result privacy as requirements are intertwined. Moreover, we prove the security of the collusions between Agg1 and the data owner and the collusions between the data customer and Agg2 in Appendix A.

6 PERFORMANCE EVALUATION

We propose to evaluate the performance of PRIDA using a use case scenario whereby multiple data customers (DCs) acting as pharmaceutical companies wish to discover the side effects of some medicine on multiple patients who take the underlying medicine, using some statistics such as sum, average, etc. over this target group. In this scenario, when patients acting as data owners (DOs) are willing to participate in this discovery with their private data, patients use threshold FHE and 2PC to protect their data before sending them to the non-colluding two aggregators. Once the aggregators receive these protected data, they jointly perform the required operations over these data and send the encrypted aggregate result to the authorized pharmaceutical companies if chosen by enough patients. Later on, these companies decrypt the result. Therefore, we conduct several experiments with different numbers of DOs and DCs and analyze the computational cost at each party.

Computational and communication complexity.

We have worked on the computational and communication costs of PRIDA as presented in Table 2. Remark that the complexities of basic operations and representation are assumed to be negligible, i.e., $O(1)$.

Computational cost for DO: $O(\ell^2)$. The computation for a data owner starts with (i) pseudorandom generation for α and β ($O(\ell)$ complexity) ; (ii) multiplication of α and β to compute γ ($O(\ell^2)$ complexity); and (iii) secret sharing for d, c, α, β, γ ($O(\ell)$ complexity).
Communication cost for DO: $O(\ell)$. A data owner constructs only one communication with Aggregator1 (Note that Aggregator2 receives the data coming from DO through Aggregator1) and sending the shares of d, c, α, β, γ takes $O(\ell)$ complexity.

Computational cost for Aggregator1 (or Aggregator2): $O(\ell^2)$. Aggregator1 (i) performs multiplication in key generation in $O(\ell^2)$ complexity and (ii) simply computes $c[d]$ for all DOs and sums all latter multiplication over 2PC in $O(\ell^2)$ complexity.
Communication cost for Aggregator1 (or Aggregator2): $O(\ell)$. Aggregator1 interactively computes the public key and the aggregation operation in $O(\ell)$ complexity.

Computational cost for DC: $O(\ell^2)$. A data customer calculates multiplication in key generation in $O(\ell^2)$ complexity.
Communication cost for DC: $O(\ell)$. A data customer interactively computes the public key in $O(\ell)$ complexity.

Prototype Implementation: Experimental Setup.

We have also implemented PRIDA based on Th-FHE and 2PC with the PALISADE library v1.11.9 [59]. We have employed the Th-FHE schemes, namely BFV and CKKS, to implement PRIDA. We have followed the standard HE security recommendations (e.g., 128-bit security) indicated in [3] for PRIDA. We implemented the 2PC protocol on our own: Data is shared by generating a random number first and then computing the other share accordingly: Additions of shares are performed locally, and multiplications are implemented according to the Beaver triplets algorithm [8]. All experiments have been carried out using a desktop computer with a 3.5GHz Intel Core i7-7800X processor, 128GB RAM for DO and DC, and a server computer with dual 2.50GHz Intel Xeon E7-8890v3 processor, 1.97TB RAM for Aggregator1 and Aggregator2.

Prototype Implementation: Timing results.

We have first evaluated the computation cost of each PRIDA player in a scenario with 100 DOs and 1 DC for PRIDA. The results per party are shown in Table 4. These values correspond to an average of measurements from ten executions. We first observe that only aggregators in PRIDA perform costly operations. Indeed, while Aggregator2 takes 1.22 seconds with Th-CKKS and 1.51 seconds with Th-BFV, Aggregator1 takes 1.17 and 1.48 seconds, respectively. We observe a slight difference in the computation time between Aggregator1 and Aggregator2 because this difference originates from the different workloads attributed to each aggregator: Aggregator2, indeed, performs additional operations to finalize the aggregation phase (see Algorithm 1).

6.1 Comparison with the state-of-the-art

We have also studied the comparison of the state-of-the-art solutions in Table 2: PRIDA supporting **multiple data customers**, **user privacy**, and **data privacy** is more realistic than existing solutions. As many prior works do not meet these privacy goals, we compare our proposal, PRIDA, particularly, with DP-based (with SSS [69]) designs [10 13] and MPC-based designs [1 24]. Before the comparison for the computation and communication complexity, it is essential to remark that PRIDA provides anonymity of data owners, data control of data owners over data customers, and collusion resistance, which are not guaranteed in [1 10 13 24] (See Table 1).

Table 2: Comparison for computational and communication complexity of PRIDA and the state-of-the art

		Ref.	*Data Owner*	*Data Customer*	*Aggregator1*	*Aggregator2*
Computation	*DP-based*	[13]	$O(n^2 + \ell n)$	NA	$O(n^2 \ell)$	NA
		[10]	$O(\log^2 n + \ell \log n)$	NA	$O(n(\log^2 n + \ell \log n))$	NA
	MPC-based	[24]	$O(M \log M)$	NA	$O(n(M \log M))$	NA
		[1]	$O(\ell)$	NA	$O(\ell)$	$O(\ell)$
		PRIDA	$O(\ell^2)$	$O(\ell^2)$	$O(\ell^2)$	$O(\ell^2)$
Communication	*DP-based*	[13]	$O(n + \ell)$	NA	$O(n^2 + \ell n)$	NA
		[10]	$O(\log^2 n + \ell)$	NA	$O(n(\log^2 n + \ell))$	NA
	MPC-based	[24]	NA	NA	$O(n)$	NA
		[1]	$O(\ell)$	NA	$O(n + \ell^2)$	$O(n + \ell^2)$
		PRIDA	$O(\ell)$	$O(\ell)$	$O(\ell)$	$O(\ell)$

NA*: Not Applicable,* n*: #DOs,* ℓ*: bit size in MPC,* M*: #Multiplications*

As shown in Table 2, data owners in PRIDA, unlike [10 13], do not need to compute many operations and interact with each other to achieve their input privacy. Data owners further store the keys and secret shares in Bonawitz et al. [13] while suffering from user privacy features like *data control* over data customers. Even though Bell et al. [10] improve the computational and communication costs of [13] for users, it still requires users to compute many operations and communicate with each other. In addition, what PRIDA provides, like aggregated result privacy and multiple data customers, does not seem to be possible in [10 13].

Prio [24] does not consider the data customer as an additional party interested in receiving the aggregated result. Indeed, data owners act as data customers. Prio enables each data owner to send its Arithmetic shares to multiple aggregators. Then aggregators sum these shares locally and send back the aggregated result to data owners, and finally, they obtain the aggregated result. Authors present the computation complexity in terms of M, which denotes the number of multiplication gates needed for verification with a lower bound of 2^{10}, and a limitation for the size of the field where they define aggregation operations over DOs' private data. On the other hand, Prio+ [1] employs Boolean shares to improve the verification performance result in [24], which is the only difference. When comparing the workload of aggregator(s) in [1 10 13 24], PRIDA involving another aggregator is still promising and reasonable as it can serve realistic use cases. Since $n > \ell$ and $M \gg \ell$, which are more realistic in real-world scenarios, PRIDA surpasses the state-of-the-art in terms of computational and communication complexities. This claim is supported by the complexity and the timing evaluation presented in Table 2 and Table 3. As [1 24] does not present any timing evaluation of the aggregation process, we compare our experimental results with [10 13]. Note that these solutions lack the properties we intended in PRIDA, like anonymity, data control, and collusion resistance.

For 1000 data owners, PRIDA with Th-CKKS takes 24.24 seconds for the aggregation; [13] takes 415.47 seconds, and [10] improves 100x in DOs' cost while the cost of the aggregator in [10] takes roughly the same time as [13]. Thus, the total cost of the aggregation operation in [10] is 413.78 seconds. As observed from Table 2, the computational complexity of PRIDA is better than the state-of-the-art solutions. Indeed, PRIDA outperforms experimental time complexities of [10 13] 17x better.

Table 3: Comparison of PRIDA with state-of-the-arts (in s).

Ref.	*Data Owner*	*Data Customer*	*Aggregator(s)*	*Total*
PRIDA	NA	0.34	11.7 + 12.2	24.24
[10]	0.017	NA	413.77	413.78
[13]	1.7	NA	413.77	415.47

To summarize, we have observed that PRIDA achieves a good balance between computational and communication costs in data aggregation, particularly for data owners who gain their privacy and take less computation when compared to the state-of-the-art proposals.

6.2 PRIDA based on multi-key FHE

In order to compare even more the performance evaluation of PRIDA, which combines the use of 2PC with Th-FHE to guarantee ***user privacy*** and ***data privacy***, and to support ***multiple data customers***, we have further studied PRIDA by employing multi-key fully homomorphic encryption (FHE) with 2PC (See the details of multi-key FHE in Section B). In this section, we present the two versions of PRIDA based on asymmetric multi-key FHE (MK-FHE) and symmetric multi-key FHE (MK-TFHE), respectively, and focus more on analyzing their performance evaluation with our initial proposal, which outperforms the two new versions. For more details, we refer readers to Appendices C and D.

Discussion.

Our empirical results show that PRIDA can compute the aggregation in 2.73 seconds with Th-CKKS, which is at least 3-fold better in computational cost than the aggregation in PRIDAv2, i.e., MK-CKKS-based PRIDA, and 2000-fold more efficient in MK-TFHE-based PRIDA.

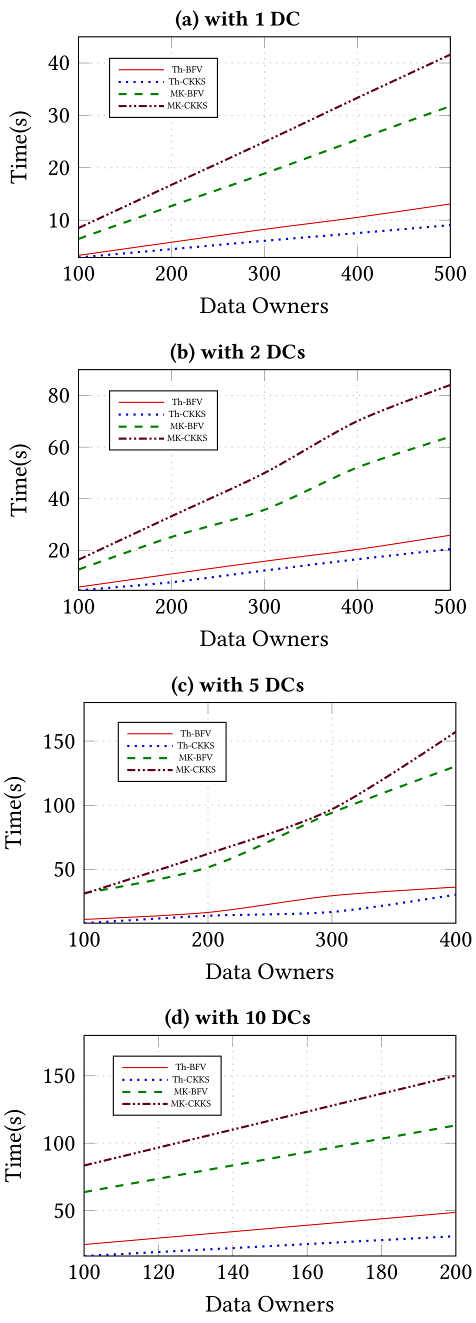
Table 4: Timing evaluation for each PRIDA player (in s). The timings of Aggregators are provided in a scenario consisting of 100 DOs and 1 DC.

Protocols	*Data Owner*	*Data Customer*	*Aggregator1*	*Aggregator2*
PRIDA with Th-BFV	**0.009**	**0.22**	**1.48**	**1.51**
PRIDAv2 with MK-BFV	0.299	0.09	3.18	3.21
PRIDA with Th-CKKS	**0.009**	**0.34**	**1.17**	**1.22**
PRIDAv2 with MK-CKKS	0.294	0.07	4.16	4.19
PRIDAv3 with MK-TFHE	4.452	2.28	2632.25	3160.70

As we have presented the performance evaluation of PRIDA, we have also analyzed the computation cost of PRIDA players for PRIDAv2 and PRIDAv3 shown in Table 4[3]. We observe that in a scenario with 100 DOs and 1 DC, Aggregator2 takes 3.21 seconds with MK-BFV and 4.19 seconds with MK-CKKS within PRIDAv2, while Aggregator1 takes 3.18 and 4.16 seconds, respectively. While Aggregator2 takes 3160.71 seconds (with MK-TFHE) and Aggregator1 takes 2632.25 seconds, data owner and data customer take 4.45 and 2.27 seconds, respectively (For this result, we have used the desktop computer). As observed that PRIDAv3 takes more time than the two other protocols, the difference between the results of the symmetric and asymmetric versions is nonnegligible. In order to explain the slowness of PRIDAv3, we refer readers to the values in Table 3 in [19]: 1 NAND operation takes time between 1.90 and 7.16 seconds according to the number of keys varying between 4 and 8, respectively. Since PRIDAv3 contains multiple NAND and other homomorphic operations with 5 keys, our results are valid and overlap with the results of [19]. Also, we believe that the overhead is caused mainly due to the additional layers of encryption, for which the MK-TFHE library has not been optimized yet.

[3]These values correspond to an average of measurements from ten executions.

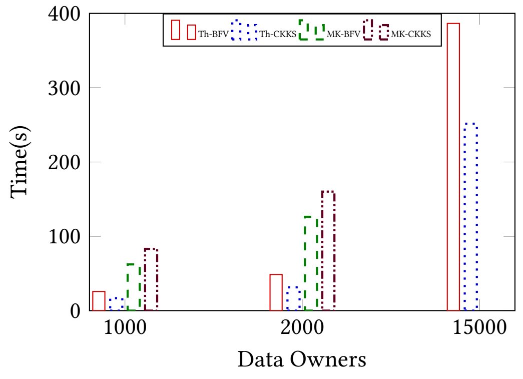
Figure 2: Data aggregation time for PRIDA (Th-FHE) and PRIDAv2 (MK-FHE) using BFV and CKKS.



In order to study the **scalability of PRIDA**, we have also measured the time of the actual aggregation operation in PRIDA (namely, the preliminary counting phase, the aggregation phase, and the decryption phase) with respect to a larger number of DOs and an increasing number of DCs. Since The PRIDAv3 cost is high and in order to make a fair assessment, we propose to study PRIDA and PRIDAv2 only. Detailed results are shown in Figure 2[3]. We remind that PRIDA enables DOs to control their data by choosing which DC can receive the aggregate result for which they are contributing. For our experiment, the anonymity threshold t is 50; yet, in general, threshold t can be considered a public parameter chosen by the two aggregators. In order to consider all possible scenarios with respect to the authorization aspects, we propose that in Figure 2b, DOs authorize each DC uniformly randomly whereas, in Figure 2c, each DO authorizes three out of five DCs (corresponding to a ratio of 50%). Lastly, in Figure 2d, DOs authorize nine DCs out of ten.

We observe that the **aggregation computation time linearly increases** with respect to the number of DOs and DCs. We also observe a slight difference between the execution times of PRIDA and PRIDAv2: PRIDAv2 relies on multiple keys, the execution time of the underlying MK-FHE . Eval algorithm linearly increases with the number of keys, and PRIDA employing Th-FHE . Eval approximately equals to Eval with one key only (i.e., the encryption algorithm uses one single public key). On the other hand, PRIDAv2 does not require an additional setup phase since, as opposed to PRIDA, each party independently generates its keying material.

Figure 3: PRIDA scalability.



In order to provide the scalability feature of PRIDA even more (to study **the PRIDA limits** without any optimizations like parallelizations, SIMD, etc.), we have run PRIDA and PRIDAv2 with 1K, 2K, and 10K data owners with 1 data customer. As depicted in Figure 3, we observe that privacy-preserving data aggregation operations remain possible: When PRIDA with Th-CKKS can support up to 15K data owners in 4.2 mins, PRIDAv2 with MK-BFV is able to perform aggregation with up to 2K data owners in 2 mins. Further, PRIDA supporting **user privacy**, **data privacy**, and **multiple data customers** is more realistic than the existing solutions and can be extended for the applications of federated learning, which requires data aggregation.

To summarize, PRIDA based on Th-FHE is more efficient than the two versions, namely PRIDAv2 and PRIDAv3: PRIDA requires the two aggregators and each data customer to jointly generate a common public key, whereas PRIDAv2 and PRIDAv3 do not need a jointly generated unique key, on the other hand, DOs need to perform more operations to protect their input privacy. Therefore, our experiments show that PRIDA achieves a good balance between computational and communication costs in privacy-preserving data aggregation.

7 CONCLUSION

This work proposes PRIDA, a privacy-preserving data aggregation solution combining threshold fully homomorphic encryption (Th-FHE) and secure two-party computation. Thanks to the use of these two cryptographic building blocks with a setting involving two non-colluding aggregators, PRIDA supports scenarios with more than one data customer. Furthermore, PRIDA enables data owners to control their data by deciding which data customers can access the resulting aggregated information. Moreover, with the introduction of an anonymous counting phase, data customers can discover the aggregation results only when a sufficient number of data owners (i.e., greater than a pre-defined threshold) authorize them. We have provided a detailed security analysis of PRIDA considering potential adversaries, including potential collusions among parties. Lastly, we instantiated PRIDA in two other different schemes, namely multi-key fully homomorphic encryption (PRIDA v2) and multi-key TFHE (PRIDA v3), to provide a detailed comparison with PRIDA based on Th-FHE: Our experimental results showed that PRIDA is more efficient than the two MK-FHE based versions of PRIDA and further when comparing the state-of-the-art solutions, PRIDA supporting user privacy and multiple data customers is promising.

REFERENCES

[1] Surya Addanki, Kevin Garbe, Eli Jaffe, Rafail Ostrovsky, and Antigoni Polychroniadou. 2022. Prio+: Privacy Preserving Aggregate Statistics via Boolean Shares. In *Security and Cryptography for Networks*, Clemente Galdi and Stanislaw Jarecki (Eds.). Springer International Publishing, Cham, 516–539.

[2] California Privacy Protection Agency. 2018. California Consumer Privacy Act. https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?lawCode=CIV&division=3.&title=1.81.5.&part=4.&chapter=&article

[3] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. 2018. *Homomorphic Encryption Security Standard.* Technical Report. HomomorphicEncryption.org, Toronto, Canada.

[4] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. 2012. Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In *Advances in Cryptology – EUROCRYPT 2012*, David Pointcheval and Thomas Johansson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 483–501.

[5] Paris Attitude. 2018. French Medical System : An overview of Parisian Hospitals. https://blog.parisattitude.com/en/overview-parisian-hospitals-french-medical-system.

[6] Eugene Bagdasaryan, Peter Kairouz, Stefan Mellem, Adrià Gascón, Kallista Bonawitz, Deborah Estrin, and Marco Gruteser. 2022. Towards Sparse Federated Analytics: Location Heatmaps under Distributed Differential Privacy with Secure Aggregation. PoPETs. https://arxiv.org/abs/2111.02356

[7] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. 2020. Private Summation in the Multi-Message Shuffle Model. CCS. https://arxiv.org/abs/2002.00817

[8] Donald Beaver. 1991. Efficient Multiparty Protocols Using Circuit Randomization. CRYPTO.

[9] Constance Beguier, Mathieu Andreux, and Eric W. Tramel. 2021. Efficient Sparse Secure Aggregation for Federated Learning.

[10] James Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (2020).

[11] Fabrice Benhamouda, Marc Joye, and Benoit Libert. 2016. A New Framework for Privacy-Preserving Aggregation of Time-Series Data. Association for Computing Machinery.

[12] Igor Bilogrevic, Julien Freudiger, Emiliano De Cristofaro, and Ersin Uzun. 2014. What's the Gist? Privacy-Preserving Aggregation of User Profiles. ESORICS.

[13] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. CCS.

[14] Zvika Brakerski. 2012. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *Advances in Cryptology – CRYPTO 2012*, Reihaneh Safavi-Naini and Ran Canetti (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 868–886.

[15] Carlo Brunetta, Georgia Tsaloli, Bei Liang, Gustavo Banegas, and Aikaterini Mitrokotsa. 2021. Non-interactive, Secure Verifiable Aggregation for Decentralized, Privacy-Preserving Learning. In *Information Security and Privacy*, Joonsang Baek and Sushmita Ruj (Eds.). Springer International Publishing, Cham, 510–528.

[16] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. 2010. SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics. Proceedings of the 19th USENIX Conference on Security.

[17] T. Hubert Chan, Elaine Shi, and Dawn Song. 2012. Privacy-Preserving Stream Aggregation with Fault Tolerance. FC.

[18] Hao Chen, Ilaria Chillotti, and Yongsoo Song. 2019. MK-TFHE Library. https://github.com/ilachill/MK-TFHE.

[19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. 2019. Multi-Key Homomophic Encryption from TFHE. ASIACRYPT.

[20] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. 2019. Efficient Multi-Key Homomorphic Encryption with Packed Ciphertexts with Application to Oblivious Neural Network Inference. CCS.

[21] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology – ASIACRYPT 2017*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer International Publishing, Cham, 409–437.

[22] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2016. Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds. In *Advances in Cryptology – ASIACRYPT 2016*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 3–33.

[23] The European Commission. 2016. Regulation of the European Parliament and of the Council concerning the respect for private life and the protection of personal data in electronic communications and repealing Directive 2002/58/EC (Regulation on Privacy and Electronic Communications).

[24] Henry Corrigan-Gibbs and Dan Boneh. 2017. Prio: Private, Robust, and Scalable Computation of Aggregate Statistics. 14th USENIX Symposium on Networked Systems Design and Implementation.

[25] Alex Davidson, Peter Snyder, E. B. Quirk, Joseph Genereux, and Benjamin Livshits. 2022. STAR: Distributed Secret Sharing for Private Threshold Aggregation Reporting. CCS.

[26] DNYUZ. 2023. Your Health Information Was Hacked. What Now? https://dnyuz.com/2023/12/07/your-health-information-was-hacked-what-now/.

[27] Zeki Erkin. 2015. Private data aggregation with groups for smart grids in a dynamic setting using CRT. Proceedings of the 2015 IEEE International Workshop on Information Forensics and Security, WIFS.

[28] Zeki Erkin and G. Tsudik. 2012. Private computation of spatial and temporal power consumption with smart meters. Proceding International Conference of Applied Cryptography and Network Security.

[29] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptol. ePrint Arch.* 2012 (2012), 144.

[30] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad Reza Sadeghi, Thomas Schneider, Hossein Yalame, and Shaza Zeitouni. 2021. SAFELearn: Secure Aggregation for private FEderated Learning. DLS'21.

[31] David Froelicher, Patricia Egger, João Sá Sousa, Jean Louis Raisaro, Zhicong Huang, Christian Mouchet, Bryan Ford, and Jean-Pierre Hubaux. 2017. UnLynx: A Decentralized System for Privacy-Conscious Data Sharing. Proceedings on Privacy Enhancing Technologies.

[32] David Froelicher, Juan Ramón Troncoso-Pastoriza, Apostolos Pyrgelis, Sinem Sav, Joao Sa Sousa, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. 2020. Scalable Privacy-Preserving Distributed Learning. CoRR. https://arxiv.org/abs/2005.09532

[33] David Froelicher, Juan Ramón Troncoso-Pastoriza, Apostolos Pyrgelis, Sinem Sav, Joao Sa Sousa, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. 2021. Scalable Privacy-Preserving Distributed Learning. POPETS.

[34] David Froelicher, Juan Ramón Troncoso-Pastoriza, João Sá Sousa, and Jean-Pierre Hubaux. 2019. Drynx: Decentralized, Secure, Verifiable System for Statistical Queries and Machine Learning on Distributed Datasets. CoRR. http://arxiv.org/abs/1902.03785

[35] Flavio D. Garcia and Bart Jacobs. 2011. Privacy-Friendly Energy-Metering via Homomorphic Encryption. In *Security and Trust Management*, Jorge Cuellar, Javier Lopez, Gilles Barthe, and Alexander Pretschner (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 226–238.

[36] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems* 33 (2020), 16937–16947.

[37] Oded Goldreich, Silvio Micali, and Avi Wigderson. 2019. How to play any mental game, or a completeness theorem for protocols with honest majority. Providing Sound Foundations for Cryptography.

[38] IBM. [n. d.]. Cost of data breach 2022. https://www.ibm.com/reports/data-breach.

[39] Mihaela Ion, Benjamin Kreuter, Erhan Nergiz, Sarvar Patel, Mariana Raykova, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. 2020. Private Intersection-Sum Protocols with Applications to Attributing Aggregate Ad Conversions. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. 370–389. https://eprint.iacr.org/2019/723.pdf
[40] Marc Joye and Benoît Libert. 2013. A Scalable Scheme for Privacy-Preserving Aggregation of Time-Series Data. FC.
[41] Swanand Kadhe, Nived Rajaraman, Onur Ozan Koyluoglu, and Kannan Ramchandran. 2020. FastSecAgg: Scalable Secure Aggregation for Privacy-Preserving Federated Learning. CCS Workshop on Privacy-Preserving Machine Learning in Practice. https://arxiv.org/abs/2009.11248
[42] Ferhat Karakoç, Melek Önen, and Zeki Bilgin. 2021. Secure aggregation against malicious users. SACMAT.
[43] Jakub Klemsa and Melek Önen. 2022. Parallel operations over TFHE-encrypted multi-digit integers. CODASPY.
[44] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. 2011. Privacy-Friendly Aggregation for the Smart-Grid. PoPETS.
[45] Iraklis Leontiadis, Kaoutar Elkhiyaoui, and Refik Molva. 2014. Private and Dynamic Time-Series Data Aggregation with Trust Relaxation. CANS.
[46] Iraklis Leontiadis, Kaoutar Elkhiyaoui, Melek Önen, and Refik Molva. 2015. PUDA - Privacy and Unforgeability for Data Aggregation. CANS.
[47] Yongjun Li, Zhaoting Su, Jiaqi Yang, and Congjie Gao. 2020. Exploiting similarities of user friendship networks across social networks for user identification. *Information Sciences* 506 (2020), 78–98.
[48] Yehuda Lindell. 2017. *How to Simulate It – A Tutorial on the Simulation Proof Technique*. Springer International Publishing, Cham, 277–346. https://doi.org/10.1007/978-3-319-57048-8_6
[49] Adriana Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan. 2011. Cloud-Assisted Multiparty Computation from Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2011/663. https://eprint.iacr.org/2011/663.
[50] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and S Yu Philip. 2022. Privacy and robustness in federated learning: Attacks and defenses. *IEEE transactions on neural networks and learning systems* (2022).
[51] Abdul Majeed and Sungchang Lee. 2020. Anonymization techniques for privacy preserving data publishing: A comprehensive survey. *IEEE access* 9 (2020), 8512–8545.
[52] K. Mandal, G. Gong, and C. Liu. 2018. NIKE-based Fast Privacy-preserving High-dimensional Data Aggregation for Mobile Devices. Submitted to IEEE Transaction on dependable and secure computing.
[53] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (S&P)*. IEEE, 691–706.
[54] Microsoft Research. 2022. Microsoft SEAL. https://github.com/Microsoft/SEAL.
[55] Mansouri Mohamad, Melek Önen, Wafa Ben Jaballah, and Mauro Conti. 2023. SoK: Secure aggregation based on cryptographic schemes for federated Learning. PETS 2023, 23rd Privacy Enhancing Technologies Symposium, 10-14 July 2023, Lausanne, Switzerland (Hybrid Conference).
[56] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (S&P)*. IEEE, 739–753.
[57] Thien Duc Nguyen, Phillip Rieger, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Ahmad-Reza Sadeghi, T. Schneider, and Shaza Zeitouni. 2021. FLGUARD: Secure and Private Federated Learning. *IACR Cryptol. ePrint Arch.* 2021 (2021), 25.
[58] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. EUROCRYPT.
[59] PALISADE. 2022. PALISADE Lattice Cryptography Library. https://palisade-crypto.org/.
[60] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data. ICLR.
[61] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. ICLR.
[62] The European Parliment and the Council of the European Union. 2018. European General Data Protection Regulation. https://gdpr.eu/
[63] Yuriy Polyakov. 2020. PALISADE: Introduction to Multiparty Homomorphic Encryption. https://palisade-crypto.org/wp-content/uploads/2020/10/PALISADE-10-30-MULTIPARTY.pdf.
[64] Health Insurance Portability and Accountability Act. 2000. HIPAA. https://www.hhs.gov/hipaa/for-professionals/index.html
[65] Assistance Publique. 2023. Hopitaux de Paris. https://fr.wikipedia.org/wiki/Assistance_publique_-_HÃťpitaux_de_Paris.
[66] Vibhor Rastogi and Suman Nath. 2010. Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption. Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data.
[67] Market Growth Reports. 2023. Sports Analytics Market Regional Analysis and Insights. https://www.linkedin.com/pulse/2031-sports-analytics-market-regional-analysis-insights/.
[68] Sinem Sav, Apostolos Pyrgelis, Juan RamónTroncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. 2021. POSEIDON: Privacy-Preserving Federated Neural Network Learning. NDSS. https://arxiv.org/abs/2009.00349
[69] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (nov 1979), 612–613. https://doi.org/10.1145/359168.359176
[70] Yingxia Shao, Jialin Liu, Shuyang Shi, Yuemei Zhang, and Bin Cui. 2019. Fast de-anonymization of social networks with structural information. *Data Science and Engineering* 4 (2019), 76–92.
[71] Elaine Shi, T.-H. Hubert Chan, Eleanor Gilbert Rieffel, Richard Chow, and Dawn Song. 2011. Privacy-Preserving Aggregation of Time-Series Data. NDSS.
[72] Jinhyun So, Basak Guler, and A. Salman Avestimehr. 2020. Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning. arXiv. https://arxiv.org/abs/2002.04156
[73] Georgia Tsaloli, Gustavo Banegas, and Aikaterini Mitrokotsa. 2020. Practical and Provably Secure Distributed Aggregation: Verifiable Additive Homomorphic Secret Sharing. Cryptography.
[74] Georgia Tsaloli, Bei Liang, Carlo Brunetta, Gustavo Banegas, and Aikaterini Mitrokotsa. 2021. DEVA: Decentralized, Verifiable Secure Aggregation for Privacy-Preserving Learning. Information Security: 24th International Conference, ISC 2021, Virtual Event, November 10–12, 2021, Proceedings.
[75] Lun Wang, Qi Pang, Shuai Wang, and Dawn Song. 2020. F2ED-Learning: Good Fences Make Good Neighbors. CoRR.
[76] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. 2010. A practical attack to de-anonymize social network users. In *2010 ieee symposium on security and privacy*. IEEE, 223–238.
[77] Xiangrui Xu, Pengrui Liu, Wei Wang, Hong-Liang Ma, Bin Wang, Zhen Han, and Yufei Han. 2022. CGIR: Conditional Generative Instance Reconstruction Attacks against Federated Learning. *IEEE Transactions on Dependable and Secure Computing* (2022).
[78] Dan Yin, Yiran Shen, and Chenyang Liu. 2017. Attribute couplet attacks and privacy preservation in social networks. *IEEE Access* 5 (2017), 25295–25305.
[79] Annika Zhang, Badih Ghazi, Neel Kamal, Pasin Manurangsi, and Ravi Kumar Ravikumar. 2022. Private Aggregation of Trajectories. PoPETs.
[80] Cheng Zhang, Honglu Jiang, Yawei Wang, Qin Hu, Jiguo Yu, and Xiuzhen Cheng. 2019. User identity de-anonymization based on attributes. In *Wireless Algorithms, Systems, and Applications: 14th International Conference, WASA 2019, Honolulu, HI, USA, June 24–26, 2019, Proceedings 14*. Springer, 458–469.
[81] Cheng Zhang, Shang Wu, Honglu Jiang, Yawei Wang, Jiguo Yu, and Xiuzhen Cheng. 2019. Attribute-enhanced de-anonymization of online social networks. In *Computational Data and Social Networks: 8th International Conference, CSoNet 2019, Ho Chi Minh City, Vietnam, November 18–20, 2019, Proceedings 8*. Springer, 256–267.
[82] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica. 2019. Helen: Maliciously Secure Competitive Learning for Linear Models. S&P.
[83] Mingxun Zhou, Tianhao Wang, T.-H. Hubert Chan, Giulia Fanti, and Elaine Shi. 2022. Locally Differentially Private Sparse Vector Aggregation. S&P. https://arxiv.org/abs/2112.03449
[84] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).

A SECURITY ANALYSIS: CONT'ED

This section incrementally studies the security analysis by introducing several definitions and further investigating potential collisions between players. Note that PRIDA aims to compute privacy-preserving data aggregation in the honest-but-curious adversarial model as mentioned in Section 2.

We introduce a few definitions [48] used in the proof of Theorem 5.1 in Section 5.

Definition A.1 (Computational Indistinguishability)**.** Let $X = X(\mathrm{m}, \lambda)$ and $Y = Y(\mathrm{m}, \lambda)$ be two probability ensembles where $\mathrm{m} \in \{0,1\}^*$ is the input of the parties and λ is the security parameter in Λ. X and Y are *computationally indistinguishable*, i.e., $X \overset{c}{\equiv} Y$, if there exists a negligible function $\mu(\lambda)$ for every non-uniform polynomial time algorithm D such that for every $\mathrm{m} \in \{0,1\}^*$ and every $\lambda \in \Lambda$,

$$|\Pr[D(X(\mathrm{m},\lambda))=1] - \Pr[D(Y(\mathrm{m},\lambda))=1]| \le \mu(\lambda). \quad (9)$$

Definition A.2 (Non-uniformity). Since $X = X(\mathrm{m},\lambda)$ and $Y = Y(\mathrm{m},\lambda)$ are computationally indistinguishable, X and Y are inherently *non-uniform*. In order to provide their non-uniformity, we need to show that X and Y are *not computationally indistinguishable* without using the negligible function $\mu(\lambda)$. When we rewrite computational indistinguishability $X \overset{c}{\equiv} Y$: For every non-uniform polynomial time algorithm D and every polynomial $p(\lambda)$, there exists $\lambda \in \Lambda$ such that for all $\lambda > \Lambda$ and every $\mathrm{m} \in \{0,1\}^*$,

$$|\Pr[D(X(\mathrm{m},\lambda))=1] - \Pr[D(Y(\mathrm{m},\lambda))=1]| < \frac{1}{p(\lambda)}. \quad (10)$$

We can write the contradiction of (A.2) as follows: There exists such D and $p(\lambda)$ such that for all $\lambda \in \Lambda$, there exists a $\lambda > \Lambda$ and m for which

$$|\Pr[D(X(\mathrm{m},\lambda))=1] - \Pr[D(Y(\mathrm{m},\lambda))=1]| \ge \frac{1}{p(\lambda)}. \quad (11)$$

For all such λ, there can be a different $\mathrm{m} \in \{0,1\}^*$. To be able to make the simulation non-uniform inherently, we need to give the value $\mathrm{m} \in \{0,1\}^*$ associated with λ.

Definition A.3 (Semantic Security). Let $\epsilon = (G, E, D)$ be an encryption scheme. ϵ is *computationally indistinguishable* if there exists a non-uniform probabilistic polynomial time algorithm $\mathcal{A}'$ for every non-uniform probabilistic polynomial time algorithm $\mathcal{A}$ such that for every probability ensemble $\{X_\lambda\}_{\lambda\in\Lambda}$ with $|X_\lambda| \le \mathrm{poly}(\lambda)$, every polynomially bounded function pair $f, h : \{0,1\}^* \to \{0,1\}^*$, every polynomial $p(\lambda)$ and sufficiently large λ

$$\Pr\left[\mathcal{A}(1^\lambda, E_k(X_\lambda), 1^{|X_\lambda|}, h(1^\lambda, X_\lambda)) = f(1^\lambda, X_\lambda)\right]$$
$$< \Pr\left[\mathcal{A}'(1^\lambda, 1^{|X_\lambda|}, h(1^\lambda, X_\lambda)) = f(1^\lambda, X_\lambda)\right] + \frac{1}{p(\lambda)} \quad (12)$$

where $k \leftarrow G(1^\lambda)$ and the probability Pr is taken over X_λ and also over the internal coin tosses of the encryption scheme algorithms G, E, and $\mathcal{A}$ or $\mathcal{A}'$. The adversary $\mathcal{A}$ has the information $E_k(X_\lambda)$ and also the auxiliary information $h(1^\lambda, X_\lambda)$ and tries to guess the value of $f(1^\lambda, X_\lambda)$ while the adversary $\mathcal{A}'$ has only the length of $(X_\lambda$ and also $h(1^\lambda, X_\lambda)$ and tries to guess $f(1^\lambda, X_\lambda)$. Note that the parties in PRIDA are honest-but-curious, and thus we do not have any auxiliary information $h(1^\lambda, X_\lambda)$. The security requirement implies that $\mathcal{A}'$ having almost nothing about the ciphertext, can correctly find $f(1^\lambda, X_\lambda)$ with approximately the same probability as $\mathcal{A}$ having the ciphertext $E_k(X_\lambda)$ for any f. In other words, whatever $\mathcal{A}$ obtains can be obtained by $\mathcal{A}'$. In the definition, we did not mention the simulation or the ideal-real world; however, the definition is in line with the paradigm [48]: In the world of $\mathcal{A}'$ holding only the length of the plaintext is the ideal world where $\mathcal{A}'$ can learn only from the plaintext length, and the proof that $\mathcal{A}'$ can obtain as much information as $\mathcal{A}$ can learn, is the comparison between the real and ideal worlds. The proof technique is to show the scheme is secure in terms of the threat model of the scheme.

Definition A.4 (Definition of Security). Let Agg1 and Agg2 be two aggregators who would like to run a protocol π, namely data aggregation, on shared and shared-encrypted inputs $x_1 = \{\langle \mathbf{cv}_i \rangle_1, [\langle \mathbf{dv}_i \rangle_1]\}$ and $x_2 = \{\langle \mathbf{cv}_i \rangle_2, [\langle \mathbf{dv}_i \rangle_2]\}$ sent by DOs to compute a functionality $f(x_1, x_2)$, simply one multiplication and two additions, which outputs $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ for each aggregator. In the execution of π, the view of aggregators are

$$\mathbf{view}_1^\pi(x_1, x_2, \lambda) = (x_1, r_1; \mathrm{m}_1, \mathrm{m}_2, \cdots, \mathrm{m}_n), \quad (13)$$
$$\mathbf{view}_2^\pi(x_1, x_2, \lambda) = (x_2, r_2; \mathrm{m}_1, \mathrm{m}_2, \cdots, \mathrm{m}_n), \quad (14)$$

where r_1, r_2 are the randomness of aggregators, λ is the security parameter, and m_i's are the intermediary messages received by aggregators. The output of π is

$$\mathbf{output}^\pi(x_1, x_2, \lambda) = (\mathbf{output}_1^\pi(x_1, x_2, \lambda), \mathbf{output}_2^\pi(x_1, x_2, \lambda)), \quad (15)$$

where $\mathbf{output}_1^\pi(x_1, x_2, \lambda)$ and $\mathbf{output}_2^\pi(x_1, x_2, \lambda)$ are the local outputs of Agg1 and Agg2. We say that π securely computes $f(x_1, x_2)$ in the presence of static honest-but-curious, non-adaptive, computationally bounded adversaries if there exist probabilistic polynomial-time simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ such that

$$\left\{\mathcal{S}_1(1^\lambda, x_1, f_1(x_1, x_2)), f(x_1, x_2)\right\} \overset{c}{\equiv} \left\{\mathbf{view}_1^\pi(x_1, x_2, \lambda), \mathbf{output}^\pi(x_1, x_2, \lambda)\right\}, \quad (16)$$

$$\left\{\mathcal{S}_2(1^\lambda, x_2, f_2(x_1, x_2)), f(x_1, x_2)\right\} \overset{c}{\equiv} \left\{\mathbf{view}_2^\pi(x_1, x_2, \lambda), \mathbf{output}^\pi(x_1, x_2, \lambda)\right\}. \quad (17)$$

We now discuss the collusions between the data owner and Aggregator1, and the collusions between the data customer and Aggregator2.

Input and output privacy against collusions between Data Owner and Aggregator. The collusions of Agg1 with one DO only result in discovering the individual input of the actual DO and the other secret shared values. Hence all remaining information is either secret shared, or shared-encrypted, or multi-key encrypted with two keys for which they do not have the corresponding secret keys.

The collusion of DO$_i$ and Agg1 is corrupted by $\mathcal{A}$: Simulator $\mathcal{S}_3$ is given the input and the output of DO$_1$ (without loss of generality, $i = 1$) and Agg1, which are $x_1 = \{\mathbf{cv}_1, \mathbf{dv}_1, \boldsymbol{\alpha}_1, \boldsymbol{\beta}_1, \boldsymbol{\gamma}_1\}$ (for DO$_1$), $x_{i1} = \{\langle \mathbf{cv}_i \rangle_1, [\langle \mathbf{dv}_i \rangle_1], \langle \boldsymbol{\alpha}_i \rangle_1, \langle \boldsymbol{\beta}_i \rangle_1, \langle \boldsymbol{\gamma}_i \rangle_1\}$ (for Agg1), $\perp$, and the security parameter λ. In the simulation, we need to show that $\mathcal{S}_3$ can generate the view of incoming messages to both parties, namely DO$_1$ and Agg1, which are $[\langle \boldsymbol{\epsilon}_1 \rangle_2]$ and $\langle \boldsymbol{\delta}_1 \rangle_2$. Note that DO$_1$ is responsible only for its data protection and sending them to aggregators; yet, the two parties can try to learn any information about the input of DO$_i$ (where $i \ne 1$), the output of DC$_j$, the anonymity of DO$_i$, and the control of DO$_i$ on DC$_j$. Formally, $\mathcal{S}_3$ works as follows:

1. $\mathcal{S}_3$ chooses a uniformly distributed random tape, r_3'.
2. $\mathcal{S}_3$ guesses intermediate messages $[\langle \boldsymbol{\epsilon}_i' \rangle_2]$ and $\langle \boldsymbol{\delta}_i' \rangle_2$ using random tape r_3'.
3. $\mathcal{S}_3$ outputs $[\langle \boldsymbol{\epsilon}_i' \rangle_2]$ and $\langle \boldsymbol{\delta}_i' \rangle_2$.

The view of DO$_1$ and Agg1, $\mathbf{view}_1^\pi(x_1, x_{i1}, x_{i2})$, in the real world is

$$\left(x_1, \langle \mathbf{cv}_i \rangle_1, [\langle \mathbf{dv}_i \rangle_1], \langle \boldsymbol{\alpha}_i \rangle_1, \langle \boldsymbol{\beta}_i \rangle_1, \langle \boldsymbol{\gamma}_i \rangle_1, r_3; [\langle \boldsymbol{\epsilon}_i \rangle_2], \langle \boldsymbol{\delta}_i \rangle_2\right), \quad (18)$$

while the view generated by the simulator, $\mathcal{S}_3(1^\lambda, x_1, x_{i1})$,

$$\left(x_1, \langle \mathbf{cv}_i \rangle_1, [\langle \mathbf{dv}_i \rangle_1], \langle \boldsymbol{\alpha}_i \rangle_1, \langle \boldsymbol{\beta}_i \rangle_1, \langle \boldsymbol{\gamma}_i \rangle_1, r'_3; [\langle \boldsymbol{\epsilon}'_i \rangle_2], \langle \boldsymbol{\delta}'_i \rangle_2\right). \tag{19}$$

As previously discussed, $\mathcal{S}_3$ and $\mathcal{S}_1$ have almost the same information. The information of DO_1 (i.e., only its input data) does not give any advantages to obtaining information about the input of DO_i, the output of DC_j. The control of DO_i on DC_j since $\mathcal{S}_3$ does not have access to the decryption key sk, it cannot simulate MP-FHE. Decrypt and also cannot simulate AS. Share. On the other hand, $\mathcal{S}_3$ can generate the intermediary messages $\langle \boldsymbol{\alpha}'_i \rangle_1, \langle \boldsymbol{\beta}'_i \rangle_1, \langle \boldsymbol{\gamma}'_i \rangle_1$ only if r'_3 is uniformly sampled from r_3. They can learn only the identity of DO_i, yet, they cannot prove whether DO_i really joins the aggregation by sending its exact data to the analytics or to whom gets its data analytics result.

Collusions between Data Customer and Aggregator. When an authorized DC and Agg2 collude, they do not discover any leakage regarding inputs of DOs and/or outputs of other DCs. Indeed, even if the Agg2 aggregates the result for an unauthorized DC and forwards it, the only information that DC would be able to decrypt would be bogus, r (DOs encrypt bogus data for unauthorized DCs).

The collusion of DC_j and Agg2 is corrupted by $\mathcal{A}$: Different from the collusion between Agg1 and DO_i, Agg2 and DC_j perform one last addition to obtain aggregate result $[s_j]$ for an authorized DC. However, this does not affect the security of protocol π since Simulator $\mathcal{S}_4$ would work on the same simulation with an additional step for the aggregate result as Simulator $\mathcal{S}_2$ does. We do not provide the proof for $\mathcal{S}_4$ for collusions between DC_j and Agg2, which perform the aggregation functionality over 2PC-secure and 2PC+MP-FHE-secure data. The two parties cannot differentiate shared-encrypted data from the random-tape data.

Control of Data Owners on Data Customers. With a choice vector, $\mathbf{cv}$, each DO can easily decide which DC can receive its data while computing the data aggregation. DO secret shares its choice vector $\mathbf{cv}$ into two and sends one share to one aggregator. Thanks to *the preliminary counting phase*, the non-authorized DC cannot receive the aggregation result. Moreover, even if a non-authorized DC colludes with Agg2, they cannot retrieve the aggregated result because Agg1 has already terminated the data aggregation for this non-authorized DC. Further, the actual data aggregation result remains bogus (i.e., a random number r is used) since bogus data is assigned to this specific element when $\mathrm{cv}_{ij} = 0$. Moreover, thanks to MP-FHE, Agg2 and DC need Agg1 to partially decrypt the encrypted aggregate result during *the decryption phase*.

Anonymity of Data Owners. A data customer DC_j can discover the aggregated result without learning the individual DO if and only if the number of DOs should be at least a pre-defined threshold t during *the preliminary counting phase*. Otherwise, DC learns nothing.

B MULTI-KEY FHE

Due to the existence of multiple DCs and the two aggregators and also comparing PRIDA based on Th-FHE, we propose to employ a particular homomorphic encryption scheme that enables one or multiple parties to encrypt or decrypt data under multiple keys, which is called multi-key fully homomorphic encryption.

To construct this building block, PRIDA revisits two different homomorphic encryption schemes, namely: (i) asymmetric multi-key FHE (MK-FHE), whereby each party encrypts its data with its own public key, and operations are performed over multiple data encrypted with multiple keys; and (ii) symmetric multi-key FHE, MK-TFHE, which is similar to the previous scheme except that the encryption key is symmetric.

Asymmetric multi-key FHE (MK-FHE). Chen et al. [20] propose multi-key variants of BFV [14 29] and CKKS [21]. Basically, multiple parties individually encrypt their private data using their own public keys, and further homomorphic operations are performed over all these data. All parties contribute to the decryption of the output. A semantically secure MK-FHE scheme is defined by seven probabilistic polynomial-time (PPT) algorithms.

- $pp \leftarrow \mathrm{MK\text{-}FHE\,.\,Setup}(1^\lambda)$: Given the security parameter $\lambda \in \Lambda$, this algorithm outputs public parameters pp.
- $(sk, pk) \leftarrow \mathrm{MK\text{-}FHE\,.\,KeyGen}(pp)$: This randomized algorithm takes the public parameters pp as input and outputs a pair of secret and public keys (sk, pk).
- $\mathrm{ct} \leftarrow \mathrm{MK\text{-}FHE\,.\,Encrypt}(\mathrm{m}, pk)$: This randomized algorithm encrypts message m with public key pk and returns a ciphertext $\mathrm{ct} = (c_0, c_1)$.
- $(\mathrm{ct}^*, T^*) \leftarrow \mathrm{MK\text{-}FHE\,.\,Pre\text{-}process}(\bar{\mathrm{ct}} = (c_0, \ldots, c_{k_i}),$ $T = \{id_1, \ldots, id_{k_i}\})$: This algorithm basically extends the input ciphertext with additional 0's in order to be able to perform the homomorphic operation over all the underlying k keys according to the mapping defined in $T^* = \{id_1, ..., id_k\}$. Hence, the algorithm returns $\mathrm{ct}^* = (c^*_0, \ldots, c^*_k)$ such that $c^*_0 = c_0$ and $c^*_i = \begin{cases} c_j & \text{if } i = id_j \text{ for some } 1 \le j \le k_i, \\ 0 & \text{otherwise.} \end{cases}$
- $\mathrm{ct}^* \leftarrow \mathrm{MK\text{-}FHE\,.\,Eval}(C, (\mathrm{ct}^*_1, \ldots, \mathrm{ct}^*_l))$: This algorithm evaluates circuit C over the l ciphertexts encrypted with multiple keys.
- $\mu_i \leftarrow \mathrm{MK\text{-}FHE\,.\,PartialDecrypt}(c^*_i, sk_{id_i})$: This algorithm takes as input c^*_i from ciphertext ct^* corresponding to the party holding secret key sk_{id_i} and outputs a partially decrypted information μ_i.
- $\mathrm{m}' \leftarrow \mathrm{MK\text{-}FHE\,.\,Merge}(c^*_0, \mu_1, \ldots, \mu_k)$: This algorithm takes as input all the partial decryptions derived from a ciphertext ct^* and outputs the final plaintext m'.

We refer readers to [20] for more details on asymmetric MK-FHE.

Symmetric multi-key FHE (MK-TFHE). Chen et al. [19] proposed the multi-key version of TFHE, whereby each party uses its symmetric secret key to encrypt and decrypt the data. The MK-TFHE scheme is defined by seven PPT algorithms:

- $pp \leftarrow \mathrm{MK\text{-}TFHE\,.\,Setup}(1^\lambda)$: This algorithm outputs public parameters pp given security parameter $\lambda \in \Lambda$.
- $sk \leftarrow \mathrm{MK\text{-}TFHE\,.\,KeyGen}(pp)$: This randomized algorithm generates secret key sk given the public parameters pp.
- $\mathrm{ct} = (b, a) \leftarrow \mathrm{MK\text{-}TFHE\,.\,Encrypt}(\mathrm{m}, sk)$: This randomized algorithm encrypts message m with secret key sk and outputs ciphertext $\mathrm{ct} = (b, a)$.

- $(\mathrm{ct}^*, T^*) \leftarrow$ MK-TFHE . Pre-process$(\bar{\mathrm{ct}} = (b, a_1, \dots, a_{k_i}),$ $T = \{id_1, \dots, id_{k_i}\})$: Similar to the case of MK-FHE, this algorithm basically extends the input ciphertext with additional 0's in order to be able to perform the homomorphic operation over all the underlying k keys according to the mapping defined in T^*. Hence, the algorithm returns $\mathrm{ct}^* = (b^*, a_1^* \dots, a_k^*)$ such that
$$b^* = b \text{ and } a_i^* = \begin{cases} a_j & \text{if } i = id_j \text{ for some } 1 \le j \le k_i, \\ 0 & \text{otherwise.} \end{cases}$$
- $\mathrm{ct}^* \leftarrow$ MK-TFHE . Eval$(C, (\mathrm{ct}_1^*, \dots, \mathrm{ct}_l^*))$: This algorithm evaluates circuit C over the l ciphertexts encrypted with multiple keys.
- $\mu_i \leftarrow$ MK-TFHE . PartialDecrypt(a_i^*, sk_{id_i}): This algorithm takes as input a_i^* from ciphertext ct^* corresponding to the party holding secret key sk_{id_i} and outputs a partially decrypted information μ_i.
- $\mathrm{m}' \leftarrow$ MK-TFHE . Merge$(b^*, \mu_1, \dots, \mu_k)$: This algorithm takes as input all the partial decryptions derived from a ciphertext ct^* and outputs the final plaintext m'.

For more details on MK-TFHE, we refer readers to [19].

We now present two versions of PRIDA, which are based on 2PC and MK-FHE (PRIDAv2) depicted in Protocol 2, and based on 2PC and MK-TFHE, namely PRIDAv3 in Protocol 3. In the next section, we will highlight the main differences between each instantiation and PRIDA based on Th-FHE.

C PRIDAV2: PRIDA WITH MK-FHE

This section presents the details of PRIDAv2 in Protocol 2, and the main difference with Protocol 1 resulting from the use of MK-FHE is highlighted with the blue color.

PRIDAv2 presented in Protocol 2 combines 2PC and asymmetric multi-key fully homomorphic encryption (MK-FHE) [20]. We present a few steps while maintaining the support of our privacy goals and multiple data customers in PRIDAv2 as well: During the *data protection phase*, each DO$_i$ secret shares its data $\mathbf{dv}_i$ and uses the three public keys of Agg1, Agg2, and the actual DC$_j$ while encrypting the shares. As previously mentioned in PRIDA (See Section 4), DOs have employed one unique public key of Agg1, Agg2, and DC$_j$. Moreover, note that the given example in the original MK-FHE proposal [20] only involves two public keys: Each party encrypts its input data with its public key; however, in PRIDAv2, DOs employ the public keys of Agg1, Agg2, and DC$_j$ to encrypt their input data. Therefore, we propose the following transformation (Step (c) to (f) in Protocol 2): Before encrypting the shares, DO$_i$ once again secret shares $\langle \mathbf{dv}_i \rangle_k$ into three shares and encrypts each share with one of the three public keys; then, MK-FHE . Pre-process is called for each of these encrypted shares, and finally, these three values are summed using MK-FHE . Eval. The obtained result corresponds to the encryption of $\langle \mathbf{dv}_i \rangle_k$ with three keys, as expected. The following steps of PRIDAv2 are the same as in Protocol 1.

In order to analyze the performance of PRIDAv2, we have implemented the asymmetric MK-FHE solution described in [20] using the BFV [14 29] and CKKS [21] schemes from the SEAL library v41.1.1 [54].

Protocol 2 PRIDA based on asymmetric MK-FHE

Inputs. DO$_i$, $i \in \{1, \dots, n\}$, inputs a choice vector $\mathbf{cv}_i$ and a data vector $\mathbf{dv}_i$ of size m. Also, a pre-defined threshold t and the public parameters pp are published.

Output. If $\mathrm{cv}_{total_j} \ge t$, DC$_j$ obtains the aggregate result s_j, $j \in \{1, \dots, m\}$.

Protocol steps:

1. *Setup executed by DC$_j$, Agg1, and Agg2.*
   a. $(sk_p, pk_p) \leftarrow$ MK-FHE . KeyGen(pp) where p = DC$_j$, Agg1, and Agg2.
2. *Data protection executed by DO$_i$.*
   a. $(\langle \mathbf{cv}_i \rangle_1, \langle \mathbf{cv}_i \rangle_2) \leftarrow$ AS. Share$(2, \mathbf{cv}_i)$.
   b. $(\langle \mathbf{dv}_i \rangle_1, \langle \mathbf{dv}_i \rangle_2) \leftarrow$ AS. Share$(2, \mathbf{dv}_i)$.
   c. $(\langle \mathrm{dv}_{ij} \rangle_{k,\mathrm{DC}_j}, \langle \mathrm{dv}_{ij} \rangle_{k,\mathrm{Agg1}}, \langle \mathrm{dv}_{ij} \rangle_{k,\mathrm{Agg2}}) \leftarrow$ AS. Share$(3, \langle \mathrm{dv}_{ij} \rangle_k)$ for $k = 1, 2$.
   d. $[\langle \mathrm{dv}_{ij} \rangle_{k,p}] \leftarrow$ MK-FHE . Encrypt$(\langle \mathrm{dv}_{ij} \rangle_{k,p}, pk_p)$ for k = 1, 2 and p = DC$_j$, Agg1, and Agg2.
   e. Call MK-FHE . Pre-process with all $[\langle \mathrm{dv}_{ij} \rangle_{k,p}]$ for $T^* = \{id_{DC_j}, id_{Agg1}, id_{Agg2}\}$.
   f. $[\langle \mathrm{dv}_{ij} \rangle_k] \leftarrow$ MK-FHE . Eval $(+, ([\langle \mathrm{dv}_{ij} \rangle_{k,1}], [\langle \mathrm{dv}_{ij} \rangle_{k,2}], [\langle \mathrm{dv}_{ij} \rangle_{k,3}]))$, k = 1, 2.
   g. Generate random vectors $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$ and compute $\boldsymbol{\gamma}_i$ such that $\boldsymbol{\gamma}_i = \boldsymbol{\alpha}_i \star \boldsymbol{\beta}_i$.
   h. *Beaver's triplets.* Call AS. Share(2, .) for $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, $\boldsymbol{\gamma}_i$.
   i. Send $[\langle \mathbf{dv}_i \rangle_k]$, $\langle \mathbf{cv}_i \rangle_k$, $\langle \boldsymbol{\alpha}_i \rangle_k$, $\langle \boldsymbol{\beta}_i \rangle_k$, and $\langle \boldsymbol{\gamma}_i \rangle_k$ to Agg$k$, $k = 1, 2$.
3. *Preliminary counting executed by Agg1 and Agg2.*
   a. Agg$k$: Obtain $\langle \mathbf{cv}_{total} \rangle_k = \sum \langle \mathbf{cv}_i \rangle_k$, $k = 1, 2$.
   b. Agg$k$: Exchange $\langle \mathbf{cv}_{total} \rangle_k$ to get $\mathbf{cv}_{total}$ = $(\mathrm{cv}_{total_1}, \dots, \mathrm{cv}_{total_j}, \dots, \mathrm{cv}_{total_m})$.
   c. Agg$k$: Label DC$_j$ as authorized if $\mathrm{cv}_{total_j} \ge t$.
4. *Aggregation executed by Agg1 and Agg2.*
   a. Jointly compute $[\mathbf{s}] = [\sum \mathbf{dv}_i \star \mathbf{cv}_i]$ for each authorized DC$_j$. The details of multiplication are provided in Algorithm 1.
5. *Decryption executed by Agg1, Agg2, and DC$_j$.*
   a. Agg2: Send the aggregate result $[\mathbf{s}]$ to Agg1.
   b. Agg$k$: $\mu_k \leftarrow$ MK-FHE . PartialDecrypt, $k = 1, 2$.
   c. Agg1: Send μ_1 to Agg2.
   d. Agg2: Send μ_1 and μ_2 to the authorized DC$_j$.
   e. DC$_j$: $\mu_3 \leftarrow$ MK-FHE . PartialDecrypt.
   f. DC$_j$: Run MK-FHE . Merge with μ_1, μ_2, μ_3.

D PRIDAV3: PRIDA WITH MK-TFHE

This section shows that PRIDA also supports the symmetric multi-key FHE in Protocol 3, and we further introduce a new algorithm, namely MK-TFHE . Post-process.

The specification of PRIDAv3, executing MK-TFHE . Post-process, is provided in Protocol 3, and the main difference with Protocol 1 resulting from the use of MK-TFHE is highlighted with the blue color.

Protocol 3 PRIDA based on MK-TFHE

Inputs. Choice vector $\mathbf{cv}_i$ and data vector $\mathbf{dv}_i$ for DO_i, pre-defined threshold t, and public parameters pp.
Output. If $\mathrm{cv}_{total_j} \geq t$, DC_j obtains the aggregate result s_j, $j \in \{1, \ldots, m\}$.
Protocol steps:

1. *Setup executed by DO_i, DC_j, Agg1, and Agg2.*
   a. Generate $sk_p \leftarrow$ MK-TFHE . KeyGen(pp), $p = \mathrm{DO}_{i_1}, \mathrm{DO}_{i_2}$.
   b. Generate $sk_p \leftarrow$ MK-TFHE . KeyGen(pp), $p = \mathrm{DC}_j$, Agg1, Agg2.
2. *Data protection executed by DO_i.*
   a. $(\langle \mathbf{cv}_i \rangle_1, \langle \mathbf{cv}_i \rangle_2) \leftarrow \mathrm{AS.Share}(2, \mathbf{cv}_i)$.
   b. $(\langle \mathbf{dv}_i \rangle_1, \langle \mathbf{dv}_i \rangle_2) \leftarrow \mathrm{AS.Share}(2, \mathbf{dv}_i)$.
   c. $(\langle \mathrm{dv}_{ij} \rangle_{k,\mathrm{DC}_j}, \langle \mathrm{dv}_{ij} \rangle_{k,\mathrm{DO}_{i_1}}, \langle \mathrm{dv}_{ij} \rangle_{k,\mathrm{DO}_{i_2}}) \leftarrow \mathrm{AS.Share}(3, \langle \mathrm{dv}_{ij} \rangle_k)$ for $k = 1, 2$.
   d. $[\langle \mathrm{dv}_{ij} \rangle_{k,p}] \leftarrow$ MK-TFHE . Encrypt$(\langle \mathrm{dv}_{ij} \rangle_{k,p}, sk_p)$, $k = 1, 2$.
   e. Call MK-TFHE . Pre-process with all $[\langle \mathrm{dv}_{ij} \rangle_{k,p}]$ for $T^* = \{id_{DC_j}, id_{DO_{i_1}}, id_{DO_{i_2}}\}$.
   f. $[\langle \mathrm{dv}_{ij} \rangle_k] \leftarrow$ MK-TFHE . Eval $(+, ([\langle \mathrm{dv}_{ij} \rangle_{k,\mathrm{DC}_j}], [\langle \mathrm{dv}_{ij} \rangle_{k,\mathrm{DO}_{i_1}}], [\langle \mathrm{dv}_{ij} \rangle_{k,\mathrm{DO}_{i_2}}]))$.
   g. Generate random vectors $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$ and compute $\boldsymbol{\gamma}_i$ such that $\boldsymbol{\gamma}_i = \boldsymbol{\alpha}_i \star \boldsymbol{\beta}_i$.
   h. *Beaver's triplets.* Call AS.Share$(., 2)$ for $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, $\boldsymbol{\gamma}_i$.
   i. Send $[\langle \mathbf{dv}_i \rangle_k]$, $\langle \mathbf{cv}_i \rangle_k$, $\langle \boldsymbol{\alpha}_i \rangle_k$, $\langle \boldsymbol{\beta}_i \rangle_k$, and $\langle \boldsymbol{\gamma}_i \rangle_k$ to Agg$k$, $k = 1, 2$.
3. *Preliminary counting executed by Agg1 and Agg2 as shown in Protocol 1.*
4. *Aggregation executed by Agg1 and Agg2.*
   a. Agg_k: Call MK-TFHE . Post-process with the known secret keys of DO_{i_k} and Agg_k to modify $[\langle \mathbf{dv}_i \rangle_k]$ related to $sk_{\mathrm{DO}_{i_k}}$ to sk_{Agg_k}.
   b. Agg_k: Exchange samples and repeat Step (a) ending up with samples encrypted with DC_j, Agg1, Agg2.
   c. Agg_k: Jointly calculate the result $[s_j]$ jointly such that $[s_j] = [\sum \mathrm{dv}_{ij} \cdot \mathrm{cv}_{ij}]$ for authorized DC_j. The details of multiplication are provided in Algorithm 1.
5. *Decryption executed by Agg1, Agg2, and DC_j.*
   a. Agg2: Send $[\mathbf{s}] = (\ldots, [s_j], \ldots)$ to Agg1.
   b. Agg_k: $\mu_k \leftarrow$ MK-TFHE . PartialDecrypt.
   c. Agg1: Send μ_1 to Agg2.
   d. Agg2: Send μ_1 and μ_2 to the authorized DC_j.
   e. DC_j: $\mu_3 \leftarrow$ MK-TFHE . PartialDecrypt.
   f. DC_j: Run MK-TFHE . Merge with μ_1, μ_2, μ_3.

MK-TFHE is constructed as an extension of TFHE [22]. The main challenge in designing PRIDA on MK-TFHE relies on the generation and distribution of pairwise symmetric keys. A large number of parties (DOs and DCs) result in a large number of pairwise keys. We, therefore, address this problem by transforming ciphertexts encrypted with individual key sets into ciphertexts encrypted with common *group* keys. More specifically, each DC_j shares one common key k_j with all DOs, and each DO establishes one pairwise key with each aggregator: When Agg1 and Agg2 receive individual ciphertexts encrypted with different sets of keys, they partially decrypt them with the keys that they know and re-encrypt them with their unique key only known by themselves. Therefore, we propose a brand new algorithm, namely MK-TFHE . Post-process defined in Algorithm 2 to remove some keying material if one possesses that keying material. We refer the reader to Section B for the details of Algorithm 2 and Protocol 3.

We have implemented PRIDAv3 using an extended version of the MK-TFHE library ([18], an experimental prototype). We have conducted several improvements: We have implemented all binary gates except for the NAND gate, which has already been available. We have also revisited the original MK-TFHE . Encrypt and MK-TFHE . Decrypt algorithms because they require all the keys from the very beginning; thus, we prevented a significant increase in the number of keys and, consequently, the size of the ciphertext. Thus, we have split Encrypt into MK-TFHE . Encrypt and MK-TFHE . Pre-process, and Decrypt into MK-TFHE . PartialDecrypt and MK-TFHE . Merge, and implemented the new MK-TFHE . Post-process algorithm. We further have implemented some optimizations: (i) We have implemented the double-and-add algorithm for the multiplication of an MK-TFHE-encrypted number by a known scalar; and (ii) we have implemented a variant of the addition algorithm, which employs ternary logical gates, and it only requires 2 bootstrappings instead of 5, hence saving about a factor of 2.5 times [43].

Protocol 3 based on MK-TFHE, requires some additional operations to be compatible with as less keying material for ciphertexts under multi-key as possible. Thanks to MK-TFHE . Post-process, in Algorithm 2, the two aggregators do not need to execute MK-TFHE . Pre-process as double the number of DOs.

In Protocol 3, DO_i generates two secret keys for the two aggregators who decrease the number of keys related to $\langle d_{ij} \rangle_k$ by employing MK-TFHE . Post-process, and add their actual keys thanks to MK-TFHE . Pre-process. Therefore, the underlying ciphertext will contain as few keys as possible.

Algorithm 2: MK-TFHE . Post-process

```
Input: c̄t = $(b, a_1, \ldots, a_k)$, $T = \{id_1, \ldots, id_l, \ldots, id_k\}$, old
       identity $id_l$, new identity $id_l^*$, and their respective
       secret keys.
Output: $\mathrm{ct}^*$, $T^*$
1 $b' \leftarrow b - a_l \cdot sk_{id_l}$.    // noise is added in the next step
2 $(b^*, a_l^*) \leftarrow$ MK-TFHE . Encrypt$(b', sk_{id_l^*})$.
3 $\mathrm{ct}^* = (b^*, a_1, \ldots, a_l^*, \ldots, a_k)$, $T^* = \{id_1, \ldots, id_l^*, \ldots, id_k\}$.
```

For the security of the new Post-process algorithm, it is crucial to add some noise, which must be done with each partial key addition and/or removal. Note that in Algorithm 2, a fresh noise is added as a part of the MK-TFHE . Encrypt algorithm is sufficient to do that once for both key addition and removal.