Privacy-Preserving Membership Queries for Federated Anomaly Detection

Jelle Vos
Delft University of Technology
Delft, The Netherlands
J.V.Vos@tudelft.nl

Sikha Pentyala
University of Washington Tacoma
Tacoma, Washington, USA
sikha@uw.edu

Steven Golob
University of Washington Tacoma
Tacoma, Washington, USA
golobs@uw.edu

Ricardo Maia
University of Brasília
Brasília, Brazil
ricardo.jmm@gmail.com

Dean Kelley
University of Washington Tacoma
Tacoma, Washington, USA
deanak@uw.edu

Zekeriya Erkin
Delft University of Technology
Delft, The Netherlands
Z.Erkin@tudelft.nl

Martine De Cock
University of Washington Tacoma
Tacoma, Washington, USA
Ghent University
Ghent, Belgium
mdecock@uw.edu

Anderson Nascimento
University of Washington Tacoma
Tacoma, Washington, USA
andclay@uw.edu

ABSTRACT

In this work, we propose a new privacy-preserving membership query protocol that lets a centralized entity privately query datasets held by one or more other parties to check if they contain a given element. This protocol, based on elliptic curve-based ElGamal and oblivious key-value stores, ensures that those 'data-augmenting' parties only have to send their encrypted data to the centralized entity once, making the protocol particularly efficient when the centralized entity repeatedly queries the same sets of data. We apply this protocol to detect anomalies in cross-silo federations. Data anomalies across such cross-silo federations are challenging to detect because (1) the centralized entities have little knowledge of the actual users, (2) the data-augmenting entities do not have a global view of the system, and (3) privacy concerns and regulations prevent pooling all the data. Our protocol allows for anomaly detection even in strongly separated distributed systems while protecting users' privacy. Specifically, we propose a cross-silo federated architecture in which a *centralized entity* (the backbone) has labeled data to train a machine learning model for detecting anomalous instances. The other entities in the federation are *data-augmenting clients* (the user-facing entities) who collaborate with the centralized entity to extract feature values to improve the utility of the model. These feature values are computed using our privacy-preserving membership query protocol. The model can be trained with an off-the-shelf machine learning algorithm that provides differential privacy to prevent it from memorizing instances from the training data, thereby providing *output privacy*. However, it is not straightforward to also efficiently provide *input privacy*, which ensures that none of the entities in the federation ever see the data of other entities in an unencrypted form. We demonstrate the effectiveness of our approach in the financial domain, motivated by the PETs Prize Challenge, which is a collaborative effort between the US and UK governments to combat international fraudulent transactions. We show that the private queries significantly increase the precision and recall of the otherwise centralized system and argue that this improvement translates to other use cases as well.

KEYWORDS

federated learning, anomaly detection, ElGamal encryption, oblivious key-value stores, differential privacy

Proceedings on Privacy Enhancing Technologies 2024(3), 186–201
© 2024 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2024-0074

1 INTRODUCTION

Privacy-preserving membership query protocols allow a centralized entity $\mathcal{S}$ to find out whether a given element is contained in the data sets of other parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ without learning anything about the other elements in those sets. In the multi-party case, such a membership query will only return positively when the queried element is contained in the set of each other party. This work proposes a protocol to perform privacy-preserving membership queries that is particularly efficient when performing multiple parallel and sequential queries on the same sets. We refer to such queries as repeated membership queries. The key to making this efficient is to use encrypted membership query filters (EMQFs), see Section 5.1.

Repeated privacy-preserving membership queries occur in large-scale systems, which are often comprised of many user-facing entities and one or a few centralized entities serving as the backbone. Such federations arise for various reasons: to increase scalability, for political and operational reasons, or simply because that is how these systems functioned historically, and reorganizations are costly. Examples include (1) financial systems such as the global payment system orchestrated by SWIFT and distributed among banks, (2) governmental systems such as tax authorities controlled by national governments but supported by states or municipalities, and (3) health care guided by insurers while provided by many medical institutions. All these systems are prone to fraud because the centralized entities have little knowledge of the users, while the other entities do not have a global view of the system. Typical approaches that address fraud in such a federated setting come at the cost of the users' privacy, and not addressing it is not an option due to its high societal cost. Privacy-preserving membership queries allow us to address fraud while preserving users' privacy.

We demonstrate the effectiveness of our solution in the financial domain. As illustrated in Fig. 1, we assume a cross-silo federated architecture in which a *centralized entity* (the backbone) has labeled data to train a machine learning (ML) model for detecting anomalous instances. The other entities in the federation are *data-augmenting entities* (the user-facing entities), which collaborate with the centralized entity to extract feature values to improve the utility of the model. In the financial domain, the centralized entity would, for instance, be a payment network system like SWIFT that holds information about financial transactions, and the data-augmenting entities are partner banks that hold additional information about the ordering and beneficiary accounts appearing in financial transactions. The task is to use the sensitive information residing with all the entities, e.g. the payment network system and the banks, to train a ML model to detect anomalies. A large body of research has already studied ways to train such a model with *output privacy* guarantees: by training the model using differential privacy (DP) techniques, any inferences made with the model can be made public while guaranteeing plausible deniability about the existence of any specific instance in the training data. The challenge is to also provide *input privacy*, which guarantees that the information held by any data-augmenting entities and the information held by any centralized entity is not disclosed to any other entity. We propose a new cryptographic protocol tailored to this setting to efficiently and privately extract a Boolean feature indicating whether the data relating to a specific instance is consistent between the centralized entity and that of one or more data-augmenting entities.

Throughout this paper, we use $\mathcal{S}$ to denote the centralized entity and $\mathcal{P}_1, \ldots, \mathcal{P}_n$ to denote the data-augmenting entities. We assume that $\mathcal{S}$ has a training dataset in which each instance is labeled with a value denoting whether it is anomalous or not. To provide output privacy, $\mathcal{S}$ can train a ML $\mathcal{M}$ over its data with any of a variety of supervised DP model training algorithms that have been proposed in the literature, including for logistic regression, tree ensembles, and neural networks [3, 17, 18, 25]. We note that this choice of model is completely free, and we do not expand on it in this work. The emphasis in this paper is on improving the utility of $\mathcal{M}$ by extracting feature values that represent whether information held by $\mathcal{S}$ is consistent with information held by one or more of the $\mathcal{P}_i$'s, e.g. whether the information about the ordering and beneficiary accounts listed in a transaction known to a payment network system is consistent with the information known by the respective sending and receiving banks.

From a technical point of view, our private feature extraction protocol is built on top of a novel private set membership protocol that is especially efficient when performing many sequential queries. To make the protocol efficient, we instantiated the cryptosystem over an elliptic curve and implemented it in Rust (with a Python wrapper). The resulting protocol has low computational and communication demands. Moreover, typical protocols for private set membership or private set intersection based on oblivious key-value stores [29] and built using OT extension protocols [6] reveal the protocol's output in the clear. This is problematic when the result of the protocol needs to be used in other private computations. We provide an extension of our protocol that overcomes this limitation by outputting ElGamal encryptions of the data and using a custom private equality test that uses its homomorphic properties. We are not aware of similar constructions in the literature.

Motivated by the 2023 PETs prize challenge,[1] which is a collaborative effort of the US and UK governments, we demonstrate the effectiveness of our approach in the financial domain. We show that the private queries significantly increase the precision and recall of the otherwise centralized system and argue that this improvement translates to other use cases as well.

Our contributions can be summarized as follows:

- We propose a new multi-party private set membership protocol that is also of independent interest (e.g., for multi-party private set intersections), in which we formalize the concept of an EMQF.
- We propose a secure equality protocol over elliptic curve-based ElGamal ciphertexts, also of independent interest.
- To the best of our knowledge, we discuss the first concretely efficient distributed anomaly detection system in a federated star topology that preserves both input and output privacy.

After describing related work in Sec. 2 and preliminaries in Sec. 3, we provide a high level description of our solution in Sec. 4. We follow up with a detailed description for private consistency queries in Sec. 5 and a privacy proof in Sec. 6. In Sec. 7 we document experiments that demonstrate the utility and scalability of our method, which was a prize finalist in the 2023 PETs prize challenge.

2 RELATED WORK

We go over previous works that tackle similar (sub-)problems. First, we discuss previous works for performing private membership queries by studying private set intersection protocols. After that, we analyze solutions related to our application in the field of privacy-preserving federated learning, which also tackle learning in federations while providing privacy guarantees.

2.1 Private Membership Queries

Private membership queries form the basis of private set intersection protocols, which have been studied extensively. In this work, we are interested in the multi-party case, where a querier can check the membership of an element with multiple parties at once and only receive a positive result if each set contains the element. We briefly provide an overview of some of the most recent work on these protocols. We distinguish between multi-party private membership queries and multi-party private set intersections (MPSI). In the latter case, we discuss two categories: protocols based on multiple instances of two-party computation (2PC) and protocols based on homomorphic encryption (HE). Note that while some of the protocols discussed here are efficient when it comes to computation and communication, each individual query has a large cost

[1]https://www.drivendata.org/competitions/group/nist-federated-learning/

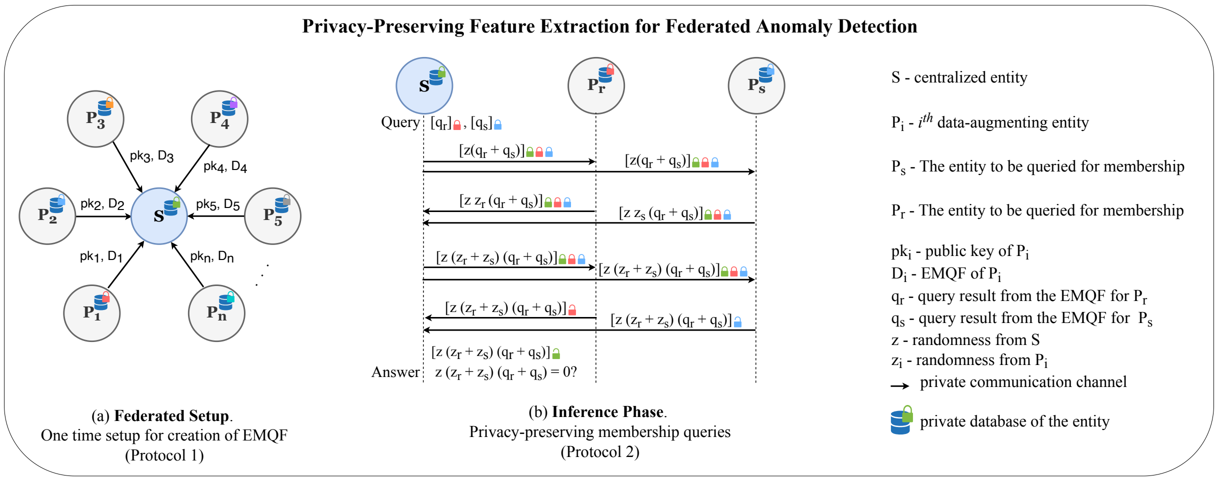
Privacy-Preserving Feature Extraction for Federated Anomaly Detection
P3
P4
P2
S
P5
P1
Pn
pk3, D3
pk4, D4
pk2, D2
pk5, D5
pk1, D1
pkn, Dn
(a) Federated Setup.
One time setup for creation of EMQF
(Protocol 1)
S
Pr
Ps
Query [qr], [qs]
[z(qr + qs)]
[z(qr + qs)]
[z zr (qr + qs)]
[z zs (qr + qs)]
[z (zr + zs) (qr + qs)]
[z (zr + zs) (qr + qs)]
[z (zr + zs) (qr + qs)]
[z (zr + zs) (qr + qs)]
[z (zr + zs) (qr + qs)]
Answer z (zr + zs) (qr + qs) = 0?
(b) Inference Phase.
Privacy-preserving membership queries
(Protocol 2)
S - centralized entity
Pi - ith data-augmenting entity
Ps - The entity to be queried for membership
Pr - The entity to be queried for membership
pki - public key of Pi
Di - EMQF of Pi
qr - query result from the EMQF for Pr
qs - query result from the EMQF for Ps
z - randomness from S
zi - randomness from Pi
private communication channel
private database of the entity

Figure 1: (a) Architecture diagram of the federation; (b) A private consistency check involving two data-augmenting clients

associated to it. In other words, these protocols are not concretely efficient for performing many queries in parallel or sequentially.

Private set intersections using 2PC. This type of MPSI protocols combines multiple executions of two-party computations that inherently perform membership queries to find the intersection of all separate sets. This closely resembles our protocol, except these protocols do not rely on homomorphic encryption. The most recent protocol in this category is that by Nevo et al. [46]. This protocol uses OKVSs encoding secret shares in combination with oblivious transfers to perform the membership checks. A similar approach was proposed before that by Garimella et al. [29]. These protocols are among the most efficient for multi-party private set intersections over large sets. Another protocol in this category is by Kavousi et al. [40], which relies on oblivious pseudo-random functions (OPRFs) as the two-party computation and functions strictly in the star topology. In this work, each party encodes their set as a garbled Bloom filter. Since OPRFs are efficient building blocks, the Bloom filter can have many bins before computations become prohibitively expensive. Note that in all three works, the party receiving the final output must perform computation and communication with the other parties scaling linearly with their set size every time the protocol is executed.

Private set intersections using HE. We discuss the three of the most recent MPSI protocols based on homomorphic encryption. All three protocols function in the star topology, which makes them suitable for in a federation, where there is a centralized entity and multiple other entities. The protocol by Bay et al. [8], and subsequently by Vos et al. [58] uses encrypted Bloom filters to perform one membership query for each of the elements in the centralized entity's set. While the former uses Paillier encryption, the latter uses elliptic curve-based ElGamal, similar to this work. These protocols are concretely efficient for small set sizes, but they are less efficient than the 2PC-based MPSI protocols as the set size grows. Hazay et al. [34] present a different protocol, in which each party other than the centralized entity represents their set as the roots of an encrypted polynomial. Performing the membership query then involves privately evaluating this polynomial on the queried element. In these three works, the centralized entity may keep the encrypted Bloom filters or encrypted polynomials locally to speed up future queries, but these methods remain efficient. For Bloom filters, this is the case because the filters must be large in size not to introduce false positives, which makes them prohibitively expensive for larger set sizes. For encrypted polynomials, each invocation requires cryptographic operations on every encrypted coefficient, which requires a great deal of computation.

Private membership queries. The protocol by Chielle et al. [19] is similar to many HE-based MPSI protocols as it is based on encrypted Bloom filters. However, Bloom filters typically need to grow large in size in order to press the false positive rate. As a result, precise queries involve a large amount of cryptographic operations. This paper uses the BFV leveled-homomorphic encryption scheme. Unfortunately, the protocol leaks information as it exposes all bits of the Bloom filter selected by the hash functions (see [58], Section 6.2). Ramezanian et al. [51] propose a more complex protocol that decreases this cost by using Cuckoo filters, which are also more common in MPSI protocols. The most recent custom protocol for performing private membership queries is that by Garg et al. [28]. This protocol provides stronger security than regular membership queries because it is hard for the party holding the queried set to fool the other party of a positive query result. However, this protocol is restricted to two parties.

In this work, we are interested in performing multiple sequential queries on the same sets. We refer to a protocol that is designed for this case as a privacy-preserving repeated membership query protocol. MPSI protocols can be naively turned into such repeated membership query protocols, but this is highly inefficient: While an MPSI protocol allows performing multiple membership queries in parallel, they are not necessarily efficient for performing multiple queries sequentially. We provide an overview incorporating the number of sequential queries as q in Table 1. Here, k represents the maximum size of the parties' sets and n is the number of involved parties. We also use t to denote the number of parties that can collude before the security guarantees no longer hold, and h to denote the number of hash functions in a Bloom filter.

In Table 1, we give two examples of MPSI protocols (Hazay et al. [34] and Vos et al. [58]) that can be easily adapted (indicated with an asterisk) so that parties only have to encode and send their set once. Their asymptotic communication complexities are identical to the protocol presented in this work. In fact, these adapted protocols can all be seen as instantiations of our protocol with a different EMQF. However, encrypted polynomials and Bloom filters are significantly larger in size and require orders of magnitude more computations to both encode and decode. We provide a concrete comparison of this in Sec. 7. Note that the extra factor n incurred in the computation of an assistant is because our work uses a different setup; the involved parties do not share a single public key.

Our solution enjoys the benefits of the HE-based MPSI protocols we discussed, running in the star topology and not having to recompute the set representation at every invocation, along with the benefits of the 2PC-based MPSI protocols, which scale well with the size of the sets. By using elliptic curve-based ElGamal rather than a partially homomorphic encryption scheme such as Paillier, the size of the OKVS stays compact, the cryptographic operations are fast to execute, and the bandwidth cost in subsequent membership queries decreases by an order of magnitude.

Table 1: Comparison of our privacy-preserving membership query protocol when expressed as a repeated multi-party private set intersection, derived from [59].

Work	**Communication**			**Computation**	
Authors	Leader	Assistant	Rounds	Leader	Assistant
Hazay [34]	$O(qnk)$	$O(qk)$	$4q$	$O(qnk)$	$O(qk)$
Inbar [36]	$O(qnkh)$	$O(qnkh)$	$3q$	$O(qnkh)$	$O(qnkh)$
Ghosh [30]	$O(qnk)$	$O(qk)$	$6q$	$O(qnk)$	$O(qk)$
Chandran [15]	$O(qnk)$	$O(qk)$	$8q$	$O(qnk)$	$O(qk)$
Garimella [29]	$O(qnk)$	$O(qk)$	$4q$	$O(qk)$	$O(qnk)$
Gordon [31]	$O(qnk + qnt)$	—	$5q$	—	—
Vos [58]	$O(qtk)$	$O(qk)$	$3q$	$O(qnkh)$	$O(qk)$
Hazay* [34]	$O(qt)$	$O(k+q)$	$1+2q$	$O(qnk)$	$O(qk)$
Vos* [58]	$O(qt)$	$O(k+q)$	$1+2q$	$O(qnkh)$	$O(qk)$
This work	$O(qt)$	$O(k+q)$	$1+2q$	$O(qnk)$	$O(qnk)$

2.2 Privacy-preserving federated learning

Federated learning (FL) [44] has emerged as a popular paradigm to train global machine learning models over data held by multiple entities, which are typically referred to as clients. Nearly all state-of-the-art FL algorithms and applications assume scenarios in which the data is horizontally partitioned, i.e. each client holds one or more instances. In the cross-silo federated architecture that we consider (see Fig. 1), the data is split both horizontally and vertically, making the mainstream FL paradigm difficult to use and analyze [39]. Orthogonal to this, it is also well understood that FL is not privacy-preserving by default, as information about the clients' training data may leak from the gradients or model parameters (see e.g. [12, 24, 39, 53]). Existing works that use a combination of privacy-enhancing technologies in the context of FL to provide end-to-end privacy address, to the best of our knowledge, only the scenario of horizontally distributed data [13, 16, 32, 37, 47, 56].

A relevant technique for cross-silo FL is secure multi-party computation (MPC), an umbrella term for cryptographic approaches that allow two or more parties to jointly compute a specified output from their private information in a distributed fashion without revealing this private information to each other [21]. While MPC typically comes with a substantial computation and communication overhead, a major advantage is that it can be readily applied to scenarios where the data is partitioned horizontally, vertically, or in any other mixed way. Nearly all of the MPC protocols proposed in the literature for model training (e.g. [4, 5, 22, 33, 41, 45, 60]) however only protect *input privacy*, i.e. they enable training of a model over data that is distributed among data holders without requiring those data holders to disclose the data. In isolation, these approaches do not provide sufficient protection if the trained model is to be made publicly known, or even if it is only made available for black-box query access, because information about the model and its training data is leaked through the ability to query the model (see e.g. [14, 26, 54, 55]). Formal privacy guarantees, in this case, can be provided by differential privacy [23]. It is, however, well known that local differential privacy, where each client adds noise to their data before sending it out, causes severe utility loss. In contrast, global differential privacy, where each client sends its data to a trusted curator responsible for adding the noise, introduces a single point of failure (namely the curator) and violates input privacy.

Input and output privacy can simultaneously be achieved in FL by combining MPC with differential privacy by replacing the trusted curator from the global DP paradigm with an MPC protocol (run across multiple mutually distrustful parties) to generate noise to perturb the model parameters, providing differential privacy guarantees. Existing methods leveraging this idea can handle data that is arbitrarily partitioned. However, such solutions are not directly applicable to our scenario because they assume that all feature values are readily available in the federation [49]. In our case, however, some features carry a high signal but have yet to be constructed by combining information from the centralized and data-augmenting entities.

While using a general-purpose MPC protocol for such joint feature extraction provides the necessary privacy guarantees, it requires significant computation and communication since MPC generic solutions for such tasks heavily depend on the circuit size (which is very high for our specific joint feature extraction task).

This work proposes a custom cryptographic protocol based on elliptic curve-based homomorphic ElGamal and oblivious key-value stores (OKVS) [29]. With almost all of the attention in the privacy-preserving machine learning (PPML) literature going to the model training phase, our proposal fills a critical gap concerning data preprocessing, namely private feature extraction. A recent work by Kadhe et al. [38] uses generic MPC and specifically avoids this private feature extraction step. This makes their protocol significantly more expensive in computation and communication.

We discuss three more recent works that tackle similar problems. Asif et al. [7] use a similar method to ours, where $\mathcal{S}$ performs membership queries on $\mathcal{P}_i$'s data, but the data is encoded as unencrypted Bloom filters. The authors wrongfully assume that this provides privacy to the users whose data is encoded. This form of information leakage is discussed in the work by Vos et al. ([58], Sec. 4.2.1). The work titled HyFL [61] solves the same problem but in a weaker threat model, where data is only encrypted against outside attackers, but $\mathcal{S}$ decrypts $\mathcal{P}_i$'s data and trains on it in plain text. Finally, a recent work titled Starlit [1] extracts a similar feature as in our work, except that it is computed by performing a one-time private set intersection between the $\mathcal{S}$'s data and $\mathcal{P}_s$ and $\mathcal{P}_r$'s data. This reveals significantly more private information than our protocol: $\mathcal{S}$ learns the result of performing membership queries with each row in their dataset on each individual $\mathcal{P}_i$'s dataset. This is essentially the worst-case leakage of our protocol.

3 PRELIMINARIES

We briefly discuss cryptographic primitives used in our protocols and introduce the definition of differential privacy.

Oblivious key-value stores. Conceptually, an OKVS is a dictionary that outputs a random-looking value for each key. If the key was encoded in the OKVS, then the value always corresponds to the value that went with it. We choose to work with the PaXoS [50] OKVS with the xxh3 statistical hash function [20]. Note that this hash function does not have to be cryptographically secure, as the security of OKVSs only relies on the statistical indistinguishability of the encoded values. The OKVS function $\mathsf{decode}(D, q)$ returns the value corresponding to the key q in OKVS D. Regardless if the key was encoded in the OKVS, the properties of the OKVS ensure that the returned value is indistinguishable from randomness.

Curve25519. In our protocols, we work over Curve25519 [9]. We denote the identity by $\mathcal{O}$, the scalar group by $\mathbb{Z}_q$, and the curve group by $E(\mathbb{Z}_q)$. Here, $q = 2^{255} - 19$ and $|E(\mathbb{Z}_q)| > 2^{252}$.

All parties have access to a generator G. While Curve25519 is defined as a Montgomery curve, it is birationally equivalent to a twisted Edwards curve. Unless specified, we use the twisted Edwards model. We also introduce two functions $\mathsf{ToMontgomery}(x)$ and $\mathsf{ToEdwards}(x, s)$, which switch x between the two curve models. Here, s denotes the sign of the twisted Edwards point, as the Montgomery point only contains the X-coordinate of the point.

Elligator maps. The main challenge when encoding curve points in an OKVS is that the OKVS requires those points to be indistinguishable from random bits. The typical compressed representation of curve points certainly does not satisfy this requirement. For example a compressed Montgomery point is simply its X coordinate, which must satisfy the strict curve equation. As a result, not every set of random bits is interpretable as the scalar representation of X. Instead, we use the Elligator2 map [10] to map between random-looking bits and curve points. Consequently, we work over Montgomery and twisted Edwards curves. We use the Montgomery model to apply the Elligator2 map given by the function $\psi : \mathbb{Z}_q \mapsto E(\mathbb{Z}_q)$ and its inverse ψ^{-1}. The inverse only returns a representative for half of the points in $E(\mathbb{Z}_q)$; otherwise, it returns $\perp$.

Differential privacy. A randomized algorithm $\mathcal{F}$ provides (ϵ, δ)-DP if for all pairs of neighboring datasets D and D' (i.e. datasets that differ in one entity), and for all subsets S of $\mathcal{F}$'s range:

$$\mathrm{P}(\mathcal{F}(D) \in S) \leq e^{\epsilon} \cdot \mathrm{P}(\mathcal{F}(D') \in S) + \delta \quad [23]. \tag{1}$$

The parameter $\epsilon \geq 0$ denotes the *privacy budget* or privacy loss, while $\delta \geq 0$ denotes the probability of violation of privacy, with smaller values indicating stronger privacy guarantees in both cases. In our context, D and D' are datasets with instances, and $\mathcal{F}$ is an algorithm to induce an ML model from a dataset, or a method to compute a statistic (like the mean) over a dataset. An DP algorithm $\mathcal{F}$ is usually created out of an algorithm $\mathcal{F}^*$ by adding noise that is proportional to the sensitivity of $\mathcal{F}^*$, in which the sensitivity measures the maximum impact a change in the underlying dataset can have on the output of $\mathcal{F}^*$.

4 SOLUTION OUTLINE

4.1 Threat model

We model all parties as polynomial-time Turing machines (PPT).

For the elliptic curve cryptography, we assume the decisional Diffie-Helman problem to hold over Curve25519 [9] and Elligator2 [10] to be statistically indistinguishable from randomness.

We implement authenticated and private communication channels between the centralized entity $\mathcal{S}$ and the data-augmenting clients $\mathcal{P}_1, \ldots, \mathcal{P}_n$ by using the authenticated encryption mode of operation EAX and AES, which we treat as a pseudorandom function. To do so, we predistribute symmetric keys among the respective parties. Alternatively, the parties can run Diffie-Helman key exchange protocols to establish such common keys.

We assume the adversaries to be honest-but-curious. That means they follow the protocol specifications, but try to obtain as much information as possible about private information from their inputs, messages exchanged, and internal randomness. Our protocols (Sec. 4.3) are designed to prevent adversaries from learning such information (see Sec. 6 for the proofs). We work with static adversaries. Our solutions can be generalized to stronger adversarial models (fully malicious/active adversaries) at the cost of having reduced efficiency. See Appendix B for an initial discussion.

In our solution, the centralized entity $\mathcal{S}$'s training data never leaves $\mathcal{S}$, not even in encrypted form. The privacy of the centralized entity's data is guaranteed even if the result of the classification (the predicted probability that an instance is anomalous) is made public. This privacy guarantee follows directly from the fact that the model is generated from the data with an algorithm that provides differential privacy (DP) guarantees (Sec. 4.2). This means that the probability that the algorithm generates a specific model from the data is very similar to the probability of generating that model if a particular instance had been left out of the data. The latter implies that what the model has memorized about individual instances is negligible. Obviously, if the result of the classification is not made public by $\mathcal{S}$, no information whatsoever leaks about the centralized entity's data (a result that follows from our secure distributed feature extraction protocol in Sec. 5).

In our solution, data never leaves the data-augmenting clients in plaintext form. In the feature extraction protocol in Sec. 5, the data-augmenting clients encrypt their data as ElGamal ciphertexts and encode the ciphertexts in oblivious key-value stores (OKVS), which they send to $\mathcal{S}$. The centralized entity $\mathcal{S}$ and the data-augmenting clients perform computations over this data while it is stays encrypted. At the end of the protocol, (1) the centralized entity and the data-augmenting clients can jointly decrypt the result (Sec. 4.3) and open it to $\mathcal{S}$, or (2) use a protocol extension to compute linear functions (such as generalized linear machine learning models) on the encrypted data. We use the former approach, so $\mathcal{S}$ learns one bit of information, which is 1 if any inconsistency is detected.

We do not consider side channel attacks in our proposal, but Protocols 1 and 3 have been designed using constant-time primitives, and the only variable-time operations do not reveal information about the inputs. However, we do not investigate the security of our solution against these attacks, and give no guarantees about our current implementation's resistance to them.

Moreover, our threat model does not take into account model inversion attacks. We perceive this as a separate issue, requiring its own countermeasures, as highlighted by [26].

4.2 Model training

We recall that the entities in our solution are the centralized entity $\mathcal{S}$, and the data-augmenting entities $\mathcal{P}_1, \ldots, \mathcal{P}_n$. $\mathcal{S}$ has a training dataset in which each instance is labeled whether it is an anomalous instance or not. $\mathcal{S}$ trains a classifier $\mathcal{M}$ over this training data. For a query instance $\boldsymbol{x}$, i.e. a new instance that needs to be classified as anomalous or not, the model $\mathcal{M}$ outputs a predicted probability $\mathcal{M}(\boldsymbol{x}) \in [0,1]$ that the instance is anomalous.

To prevent leakage of information from the predicted probabilities about the instances in the training data, we use an ML model training algorithm that provides differential privacy. In this way, our solution provides formal guarantees that the trained model $\mathcal{M}$, and hence predictions made with $\mathcal{M}$, are negligibly affected by the inclusion of any particular instance in the training data, thereby offering output privacy through plausible deniability [23]. A variety of (ϵ, δ)-DP ML model training algorithms have been proposed in the literature, including for logistic regression, tree ensembles, and neural networks [3, 17, 18, 25]. Our overall solution is general enough to allow for any (ϵ, δ)-DP ML model training algorithm to be used by $\mathcal{S}$ to train its model $\mathcal{M}$. In Sec. 7, we compare the performance of several DP model training algorithms.

4.3 Inference

During inference, $\mathcal{S}$ has to infer the probability to which each new instance $\boldsymbol{x}$ is anomalous. Our solution leverages information held by the data-augmenting entities $\mathcal{P}_1, \ldots, \mathcal{P}_n$ to improve the accuracy of the predictions made by model $\mathcal{M}$. To this end, $\mathcal{S}$ and the data-augmenting entities work together to perform a consistency check that yields $B(\boldsymbol{x}) = 1$ if there is any inconsistency between fields from x as known to $\mathcal{S}$ versus as known by the data-augmenting entities, and $B(\boldsymbol{x}) = 0$ if everything is consistent.

In the financial domain, $\mathcal{S}$ could be a payment network system interacting with banks $\mathcal{P}_1, \ldots, \mathcal{P}_n$ to check the validity of a financial transaction $\boldsymbol{x}$. The consistency check in this case would involve verifying that the names and addresses of the sender and receiver accounts in the transaction $\boldsymbol{x}$ as known to $\mathcal{S}$ match the names and addresses of the account holders as known by the sending bank $\mathcal{P}_s$ and receiving bank $\mathcal{P}_r$. In Sec. 5 we describe a protocol for performing such a consistency check without requiring $\mathcal{S}, \mathcal{P}_s$, or $\mathcal{P}_r$ to disclose their data to each other in an unencrypted manner. At the end of the protocol, we open the value of $B(\boldsymbol{x})$ to $\mathcal{S}$. We note that in this way, $\mathcal{S}$ learns only whether there was an inconsistency or not, and not from which field in $\boldsymbol{x}$ or from which data-augmenting client an inconsistency originated. We also note that the protocol in Sec. 5 is generic and supports any number of data-augmenting entities in a single query and not just two as illustrated in the example above. Moreover, the protocol in Sec. 5 supports any disjunction of equality constraints, i.e. it applies to any situation where one must check whether a database contains an entry (such as a single field or a combination of fields) that matches exactly.

The final inference result for $\boldsymbol{x}$ is computed by $\mathcal{S}$ as $\max(\mathcal{M}(\boldsymbol{x}), B(\boldsymbol{x}))$, in which $\mathcal{M}(\boldsymbol{x})$ denotes the probability predicted by the model trained by a DP algorithm (see Sec. 4.2) and $B(\boldsymbol{x})$ denotes the Boolean consistency feature jointly extracted by $\mathcal{S}$ and the data-augmenting clients. In the next section, we explain how to compute $B(\boldsymbol{x})$ in a privacy-preserving manner.

5 PRIVATE CONSISTENCY QUERIES

The underlying functionality of the private consistency queries is that of a private membership query. After all, we are testing whether each involved data-augmenting entity has a row in their dataset that matches our query. Suppose first that such a consistency query only involves the centralized entity $\mathcal{S}$ and one data-augmenting entity $\mathcal{P}_i$. It is easy to see that we can perform private consistency queries if we can realize a protocol that returns an encryption of zero (or rather the additive identity $\mathcal{O}$) if the membership check passes and randomness otherwise. We refer to this primitive as an encrypted membership query filter. The key insight in our protocol is that we can let each data-augmenting entity perform an expensive one-time setup that scales linearly with the size of their dataset, after which queries take a significantly shorter time.

5.1 Encrypted membership query filters

We introduce encrypted membership query filters (EMQFs), which are non-interactive encrypted filters for querying set membership. When queried, with high likelihood, an EMQF only returns an encryption of the identity when the element is encoded in it. It implements and satisfies the following functions and properties:

- $\mathsf{Build}(pk, R) \mapsto$ EMQF: Constructs an EMQF encoding R using encryption key pk.
- $\mathsf{Query}(\text{EMQF}, q) \mapsto C$: Queries the EMQF on element q returning a ciphertext.

Correctness $\Pr[\mathsf{Dec}(\mathsf{Query}(\text{EMQF}, q), sk) \neq \mathcal{O} \mid q \in R] \leq \mu$ and $\Pr[\mathsf{Dec}(\mathsf{Query}(\text{EMQF}, q), sk) = \mathcal{O} \mid q \notin R] \leq \mu$, where sk is the secret key for the pk used to build the EMQF.

Privacy $\mathsf{Build}(pk, R) \overset{c}{\equiv} \mathsf{Build}(pk, R')$ given $|R| = |R'|$.

A naive way to instantiate an EMQF using partially-homomorphic encryption is to encrypt every row of the set. Then, for a query, privately subtract every row with the query. However, this requires returning more than one ciphertext. Hazay et al. [34] implicitly provide an instantiation by encoding set elements as the roots of a compact but expensive-to-query encrypted polynomial and Vos et al. [58] encode the set as a large but cheap-to-query encrypted inverted Bloom filter. Both use partially-homomorphic encryption.

The idea of our paper is to encode a set as an oblivious key-value store (OKVS), which is both compact and cheap to query. We choose to work over an elliptic curve group to speed up computation, decrease the key size, and to minimize the size of the OKVS. As explained next, encoding these ciphertexts is not straightforward.

5.2 Encoding ciphertexts in the OKVS

The security of the oblivious key-value stores depends entirely on the requirement that its values are computationally indistinguishable from randomness. Note, however, that elliptic curve points by default are easily distinguishable from a uniformly random bitstring by checking whether it conforms to the curve equations. Instead, we rely on the Elligator2 map denoted by ψ to encode and decode curve points. In Algorithm 1 and 2 we show how to use this mapping to represent a point as a bitstring and vice versa. We refer to Sec. 3 for a description of the building blocks.

Algorithm 1 Encodes a twisted Edwards point p in 32 bytes indistinguishable from randomness.

```
1: procedure ToBytes($p$)
2:     $p_\mathcal{E} \leftarrow$ ToMontgomery($p$)
3:     $s \in_R \{+,-\}$                      ▷ Choose + or - representative for $p$
4:     $B \in \psi^{-1}(p_\mathcal{E}, s)$
5:     if $B = \perp$ then return $\perp$
6:     if FromBytes($B$) $= p$ then return $B$
7:     $B[31] \leftarrow B \vee 128$          ▷ Encode the sign in the MSB
8:     return $B$
```

Algorithm 2 Decodes a twisted Edwards point from 32 bytes B.

```
1: procedure FromBytes($B$)
2:     $s \leftarrow B[31] \gg 7$             ▷ Extract the sign from the MSB
3:     return ToEdwards($\psi(B), s$)
```

A data-augmenting entity can now safely generate an OKVS that realizes an EMQF using Protocol 1, where steps 2 and 3 represent the Build(pk, R) function. The EMQF's privacy is implied by the OKVS's obliviousness property. This protocol also generates a key pair. After that, the data-augmenting entity sends the OKVS and the public key to $\mathcal{S}$. The centralized entity generates a keypair in the same way, but it does not need to generate an OKVS.

5.3 Performing a query

Next, we explain how the centralized entity can perform a membership query with multiple data-augmenting entities. We present this in Protocol 2 for the case where each query involves two data-augmenting entities, but we note that the protocol can easily be extended to an arbitrary number of entities. This case involving one centralized entity and two data-augmenting entities resembles that of a federation where the centralized entity routes transactions between two data-augmenting entities.

Protocol 2 uses a variant of ElGamal ciphertexts to perform arithmetic under encryption. The goal of this protocol is for the centralized party to learn whether element q_i is included in both the sender's and receiver's EMQF. Putting encryption aside, the protocol starts by querying the sender's $\mathcal{P}_s$ and receiver's $\mathcal{P}_r$ EMQF on q_i, which is as simple as decoding and calling FromBytes on the OKVS in our OKVS-based EMQF. The EMQF of $\mathcal{P}_i$ outputs an encryption of the identity O when the query matches a row in $\mathcal{P}_i$'s database, otherwise the resulting ciphertext encrypts a random curve point. In steps 2–4 of the protocol, the centralized entity $\mathcal{S}$ computes the sum of the OKVS outputs, lets $\mathcal{P}_s$ and $\mathcal{P}_r$ multiply the result by a random scalar, and then sums those results again. Note that if both OKVS output encrypted the identity O, then the result of these steps is still O. In steps 5–6, the entities collaboratively decrypt the result such that only $\mathcal{S}$ receives the output.

Input: Set R_i containing the rows of $\mathcal{P}_i$'s database.
Output: Public key $pk_i \in E(\mathbb{Z}_q)$, secret key $sk_i \in \mathbb{Z}_q$, OKVS D_i.

(1) $\mathcal{P}_i$ randomly generates $sk_i \in_R \mathbb{Z}_q$ and computes $pk_i \leftarrow sk_i\, G \in E(\mathbb{Z}_q)$.
(2) $\mathcal{P}_i$ generates $v_j \leftarrow$ ToBytes$(r_j\, G)\, ||\,$ToBytes$(r_j\, pk_i)$ where $r_j \in_R \mathbb{Z}_q$ for $j = 1, \ldots, |R_i|$. If ToBytes returns $\perp$, resample.
(3) $\mathcal{P}_i$ encodes OKVS D_i where the keys are the rows in R_i, and the values are v_j for $j = 1, \ldots, |R_i|$.
(4) $\mathcal{P}_i$ sends pk_i and D_i to $\mathcal{S}$.

Protocol 1: One-time setup for each data-augmenting entity: generating keys and an OKVS.

Protocol 2 does not use standard ElGamal ciphertexts, as each EMQF encodes ciphertexts related to different secret keys. Instead, in step 2, $\mathcal{S}$ creates a threshold ciphertext (a, b, c, d) that keeps separate terms for each of those secret keys. To decrypt, $\mathcal{P}_s$ must multiply a by sk_s, $\mathcal{P}_r$ must do so for b, and $\mathcal{S}$ for c. Next to that, the OKVS encode pairs of curve points in Montgomery form, while the operations in this protocol require them to be in twisted Edwards form. For these reasons, we present the protocol in terms of curve points rather than ElGamal ciphertexts. We note that whenever curve points are sent between the entities, they can be compressed to a single X coordinate for space efficiency.

Our OKVS-based EMQF is correct because the OKVS always returns an encryption of the identity for elements in the set. In the case where the element is not in the set, $\mu \approx 2^{-252}$ as explained in Sec. 6.1. This protocol, however, can be understood without consideration of the specific instantiation of the EMQF or the OKVS. In Sec. 7.4, we consider the aforementioned alternatives for EMQFs and compare them to our OKVS-based EMQF using PaXoS [50].

5.4 Keeping the output encrypted

In the protocol above, $\mathcal{S}$ receives the Boolean result of the computed feature, revealing whether an inconsistency was detected. While in Sec. 6 we provide an argument why the leaked Boolean feature is permissible with regard to the privacy of a data-augmenting entity and the users they are serving, in this section we propose an extension of our approach to further minimize the leakage. We note, however, that the extended approach does not allow our inference step (see Sec. 4.3) to output a probability score between 0 and 1. Instead, with the adjusted method, our inference step outputs 0 or 1. This is an inherent limitation of the functionality.

Input: EMQFs D_i relating to R_i and queries q_i for $i \in \{s, r\}$.
Output: True if $[q_s \in R_s] \wedge [q_r \in R_r]$, or false (high probability).

(1) $\mathcal{S}$ computes $\hat{x}_i \mid\mid \hat{y}_i \leftarrow \mathsf{decode}(D_i, q_i)$ for $i \in \{s, r\}$. It transforms them into a set of Edwards points:

$$x_s \leftarrow \mathsf{FromBytes}(\hat{x}_s)\ ,\ y_s \leftarrow \mathsf{FromBytes}(\hat{y}_s)\ ,$$
$$x_r \leftarrow \mathsf{FromBytes}(\hat{x}_r)\ ,\ y_r \leftarrow \mathsf{FromBytes}(\hat{y}_r)\ .$$

(2) $\mathcal{S}$ generates $z \in_R \mathbb{Z}_q$ and computes:

$$a \leftarrow zx_s\ ,\quad b \leftarrow zx_r\ ,\quad c \leftarrow zG\ ,\quad d \leftarrow z(y_s + y_r + pk_{\mathcal{S}})\ .$$

It sends (a, b, c, d) to all entities $\mathcal{P}_i$ for $i \in \{s, r\}$.

(3) $\mathcal{P}_i$ for $i \in \{s, r\}$ generates $z_i \in_R \mathbb{Z}_q$ and computes:

$$\hat{a}_i \leftarrow z_i a\ ,\quad \hat{b}_i \leftarrow z_i b\ ,\quad \hat{c}_i \leftarrow z_i c\ ,\quad \hat{d}_i \leftarrow z_i d\ .$$

They then send $(\hat{a}_i, \hat{b}_i, \hat{c}_i, \hat{d}_i)$ to $\mathcal{S}$.

(4) $\mathcal{S}$ computes:

$$\alpha \leftarrow \hat{a}_s + \hat{a}_r\ ,\quad \beta \leftarrow \hat{b}_s + \hat{b}_r\ ,\quad \gamma \leftarrow \hat{c}_s + \hat{c}_r\ ,\quad \delta \leftarrow \hat{d}_s + \hat{d}_r\ .$$

It sends α to $\mathcal{P}_s$ and β to $\mathcal{P}_r$.

(5) $\mathcal{P}_s$ computes $\hat{\alpha} \leftarrow sk_s\alpha$, $\mathcal{P}_r$ computes $\hat{\beta} \leftarrow sk_r\beta$, and they send $\hat{\alpha}$ and $\hat{\beta}$ to $\mathcal{S}$.

(6) $\mathcal{S}$ checks if $\delta \overset{?}{=} \hat{\alpha} + \hat{\beta} + sk_{\mathcal{S}}\,\gamma$.

Protocol 2: Checks a record's consistency between $\mathcal{S}$ and two data-augmenting entities $\mathcal{P}_s$ and $\mathcal{P}_r$.

In the extended approach, we change the inference computation from $\max(\mathcal{M}(\boldsymbol{x}), B(\boldsymbol{x}))$ to $(\mathcal{M}(\boldsymbol{x}) \geq t) \vee B(\boldsymbol{x})$ for some pre-defined threshold t. Apart from $\mathcal{M}(\boldsymbol{x}) \geq t$, the inference can be entirely computed under encryption by simply adding $\mathcal{M}(\boldsymbol{x}) \geq t$ in encrypted form to the ElGamal ciphertext in step 2 of Protocol 2.

$\mathcal{S}$ can also perform inference entirely in the encrypted domain by training a DP model on both the Boolean feature and some other features. It would do so by omitting steps 5–6 of Protocol 2 and instead performing a secure equality operation that checks if the ElGamal ciphertext encrypts the identity O, returning a ciphertext encrypting O in the positive case and G otherwise.. We present a custom secure equality protocol for this purpose in Protocol 3. Unlike typical equality protocols, it does not require full decryptions (we only check if the decryption is the identity) and no conversions to secret shares. It functions in the multi-party setting and scales linearly with the number of parties. After running this protocol, $\mathcal{S}$ can run any quantized linear model over the resulting ElGamal ciphertext (linear over the Boolean feature, the other features are plaintexts). Alternatively, $\mathcal{S}$ can collaborate with the other parties to evaluate a non-linear model by introducing more interactions. The three parties proceed to finish the protocol using steps 5–6 in Protocol 2 to decrypt the final result.

The intuitive understanding of the above protocol is that each party either adds an even or odd number of encryptions of O to the set C. So long as one party does not collude, it is not clear from the decrypted ciphertexts whether the remaining party added an even or odd number of identity encryptions. The security is determined by the number of ciphertexts k that each party adds. Let us look at a toy-sized example with two parties and $k = 5$. We disregard encryption and instead work with coins. Party $\mathcal{P}_1$ creates collection

Input: Ciphertext c encrypting either the identity O or another point in $E(\mathbb{Z}_q)$.
Output: Ciphertext c' encrypting O if c encrypted O, otherwise c' encrypts G.

(1) The first party $\mathcal{P}_1$ creates a set $C \leftarrow \{c\}$ and ciphertexts c_0 and c_1 encrypting O and G, respectively.
(2) Each party $\mathcal{P}_i$ for $i = 1, \ldots, p$, in turn, does the following:
- It flips k coins $r_{i,j} \in_R \{0, 1\}$.
- It swaps and randomizes c_0 and c_1 if $r_i = \sum_{j=1}^{k} r_{i,j} = 1 \pmod 2$, setting $c_0 \leftarrow c_{r_i}$ and $c_1 \leftarrow c_{1-r_i}$.
- For $j = 1, \ldots, k$, it appends a ciphertext encrypting O to C if $r_{i,j} = 0$. Else, a ciphertext encrypting randomness.
- It multiplies each ciphertext in C by some random scalar from $\mathbb{Z}_q$.
- It shuffles set C and sends the elements to party $\mathcal{P}_{i+1}$.

(3) Parties $\mathcal{P}_1, \ldots, \mathcal{P}_p$ collaboratively decrypt the ciphertexts in C and count the number of non-identity elements as t.
(4) The first party $\mathcal{P}_1$ outputs ciphertext $c' \leftarrow c_{t \pmod 2}$ (without decrypting it).

Protocol 3: Secure equality protocol between multiple parties $\mathcal{P}_1, \ldots, \mathcal{P}_p$ for threshold additively homomorphic encryptions of points in $E(\mathbb{Z}_q)$.

$C \leftarrow \{c\}$, containing coin c, of which we do not know if it is head or tails, which we wish to find out. Let c_0 and c_1 be coins that we know to be head H and tails T, respectively. Now, $\mathcal{P}_1$ flips k random coins that it can observe. If there is an odd number of tails, it switches c_0 and c_1, otherwise it leaves them be. Next, $\mathcal{P}_1$ adds the coins to C. At this point, C may contain $\{H, T, c, T, T, H\}$, $c_0 = T$, and $c_1 = H$. Party $\mathcal{P}_2$ performs the same process, after which we may have $C = \{T, T, H, H, T, H, T, H, c, T, H\}$, and c_0 and c_1 remain unchanged because $\mathcal{P}_2$ rolled an even number of tails. When all parties are finished, $\mathcal{P}_1$ inspects C, counting the number of tails t. It returns coin $c_{t \pmod 2}$. If $c = H$, $t = 5$ in our example, so $\mathcal{P}_1$ returns coin $c_1 = H$. If $c = T$, $t = 6$, and $\mathcal{P}_1$ returns $c_0 = T$.

We explain the correctness of this protocol in more detail in Sec. 6, along with our choice of k and its impact on the protocol's security. The key idea here is that if k is large enough, the set of 'coins' looks uniform, regardless of how c is distributed. Note that unlike in Protocol 2, this protocol becomes significantly slower when the number of involved entities grows. The reason is that the set of ciphertexts grows by k ciphertexts for each entity. For applications such as verifying bank transactions this is not a problem as they only involve three entities in the equality protocol.

6 PRIVACY ANALYSIS

At the core of feature extraction is Protocol 2. We note that, in practice, this protocol can perform multiple queries in parallel. Here, we first provide a proof of correctness of this protocol and then show that it is secure in the semi-honest model. We also provide a short security argument for the secure equality operation presented in Protocol 3. We focus on our OKVS-based EMQF but the analysis works similarly for any other EMQF.

6.1 Proof of correctness

Claim 1. *Protocol 2 returns true when $q_s \in R_s \wedge q_r \in R_r$.*

Proof. Working backwards through the protocol:

$$\begin{aligned} \delta &= sk_s\alpha + sk_r\beta + sk_{\mathcal{S}}\gamma \ , \\ \hat{d}_s + \hat{d}_r &= sk_s(\hat{a}_s + \hat{a}_r) + sk_r(\hat{b}_s + \hat{b}_r) + sk_{\mathcal{S}}(\hat{c}_s + \hat{c}_r) \ , \\ \cancel{(z_s + z_r)}d &= sk_s\cancel{(z_s + z_r)}a + sk_r\cancel{(z_s + z_r)}b + sk_{\mathcal{S}}\cancel{(z_s + z_r)}c \ , \\ \cancel{z}(y_s + y_r + pk_{\mathcal{S}}) &= sk_s\cancel{z}x_s + sk_r\cancel{z}x_r + sk_{\mathcal{S}}\cancel{z}G \ . \end{aligned}$$

Since $q_s \in R_s$ and $q_r \in R_r$, then $y_s = sk_s x_s$ and $y_r = sk_r x_r$ by the functionality of an OKVS. Moreover, the setup implies $pk_{\mathcal{S}} = sk_{\mathcal{S}}G$:

$$y_s + y_r + pk_{\mathcal{S}} = sk_s x_s + sk_r x_r + sk_{\mathcal{S}}G \ , \tag{2}$$

$$y_s + y_r + pk_{\mathcal{S}} = y_s + y_r + pk_{\mathcal{S}} \ . \tag{3}$$

Claim 2. *Protocol 2 returns false with overwhelming probability when $q_s \notin R_s \vee q_r \notin R_r$.*

Proof. From (2) it follows that:

$$y_s + y_r + pk_{\mathcal{S}} \neq sk_s x_s + sk_r x_r + sk_{\mathcal{S}}G \ ,$$

must hold with overwhelming probability. Let us assume that $q_s \notin R_s$ (the argument follows the same when $q_r \notin R_r$). Then, $x_s \neq sk_{\mathcal{S}}a$ with probability $1 - |E(\mathbb{Z}_q)|^{-1}$, where $|E(\mathbb{Z}_q)| > 2^{252}$. As a result, (3) only holds with negligible probability. □

6.2 Proof of privacy

In Protocol 1 the banks only encode an OKVS and generate an ElGamal keypair. The security of this keypair, which $\mathcal{S}$ also generates, is implied by the decisional Diffie-Hellman assumption (or more precisely, by the discrete log problem). The security of the OKVS is defined by its indistinguishability from randomness. We achieve this by encoding curve points as strings that are indistinguishable from random bytes, as proposed in Algs. 1 and 2.

Given that Protocol 2 is correct (see above) and the ideal functionality is deterministic, what remains is to show that the protocol privately computes the ideal functionality in the semi-honest model [43]. We do so by showing that there exists a simulator that given the input and output can replicate the view of a party without having access to the data of other parties. To be precise, the family of simulated views is computationally indistinguishable from those of actual protocol executions. Our protocol relies on the Diffie-Hellman assumption:

Lemma 1. *The decisional Diffie-Hellman assumption implies that, for a generator point G, unknown scalars $a, b \in_R \mathbb{Z}_q$, and random point $C \in_R E(\mathbb{Z}_q)$: $(a\,G, b\,G, ab\,G) \overset{c}{\equiv} (a\,G, b\,G, C)$.*

We provide two privacy proofs: one that proves $\mathcal{S}$'s view $\mathsf{view}_{\mathcal{S}}$ to be simulatable and one for a $\mathcal{P}_i$'s view $\mathsf{view}_{\mathcal{P}}$. We keep these proofs short. We direct the reader to the work by Vos et al. [58] for a more detailed proof of a comparable protocol. To simplify notation, we do not explicitly pass the source of randomness as an input to the simulator. For the purpose of our arguments, we consider the OKVSs D_i for $i \in Q$ to be public. Given that their contents are statistically indistinguishable from randomness, this only leaks their size. This first proof shows that the protocol remains private when $\mathcal{S}$ is corrupted.

Claim 3. *There exists a simulator $\mathsf{Sim}_{\mathcal{S}}$ for PNS in Protocol 2, s.t.:*

$$\{\mathsf{Sim}_{\mathcal{S}}(1^{\lambda}, q_s, q_r, o)\}_{q_s \in Q,\, q_r \in Q,\, o \in \{0,1\}} \overset{c}{\equiv} \{\mathsf{view}_{\mathcal{S}}(q_s, q_r, \lambda)\}_{q_s \in Q,\, q_r \in Q,\, o \in \{0,1\}} \ ,$$

for security parameter $\lambda = 128$, queries q_s and q_r from query space Q, and output o.

Proof. Function $\mathsf{view}_{\mathcal{S}}$ returns inputs q_s and q_r, output o, and all incoming messages . Simulator $\mathsf{Sim}_{\mathcal{S}}$ generates an indistinguishable view by outputting the inputs and output, and randomly sampling messages $\hat{a}_s, \hat{b}_s, \hat{c}_s, \hat{d}_s, \hat{a}_r, \hat{b}_r, \hat{c}_r, \hat{d}_r, \hat{\alpha}, \hat{\beta} \in_R E(\mathbb{Z}_q)$. These messages are indistinguishable from those received in actual executions:

- In step 3, $\hat{a}_i = z_i a = z_i p_a G$ for some p_a unknown to $\mathcal{S}$. Given Lemma 1, $\hat{a}_i$ is computationally indistinguishable from randomness, even when given $a = p_a G$ and $z_i G$ (the latter is not actually given). The same argument applies to $\hat{b}_i$, $\hat{c}_i$, and $\hat{d}_i$.
- In step 5, $\mathcal{S}$ receives $\hat{\alpha} = sk_s\alpha = sk_s p_\alpha G$ and $\hat{\beta} = sk_r\beta = sk_r p_\beta G$. Given Lemma 1, α is computationally indistinguishable from randomness, even when given $pk_r = sk_r G$ and $\beta = p_\beta G$. The same argument applies to $\hat{\beta}$. □

Next, we prove that the protocol remains private when a data-augmenting entity is corrupted.

Claim 4. *There exists a simulator $\mathsf{Sim}_{\mathcal{P}}$ for $\mathcal{P}_s$ in Protocol 2, s.t.:*

$$\{\mathsf{Sim}_{\mathcal{P}}(1^{\lambda})\} \overset{c}{\equiv} \{\mathsf{view}_{\mathcal{P}_i}(\lambda)\} \ ,$$

for security parameter $\lambda = 128$ (the data-augmenting entities do not output anything).

Proof. Function $\mathsf{view}_{\mathcal{P}_i}$ returns all incoming messages of bank $\mathcal{P}_i$. Simulator $\mathsf{Sim}_{\mathcal{P}}$ generates an indistinguishable view by randomly sampling messages $a, b, c, d, \alpha \in_R E(\mathbb{Z}_q)$. These messages are indistinguishable from those received in actual executions:

- In step 2, $\mathcal{P}_i$ receives $a = zx_s = zp_a G$, $b = zx_r = zp_b G$, $c = zG$, and $d = z(y_s + y_r + pk_{\mathcal{S}}) = zp_d G$. Given Lemma 1, a is computationally indistinguishable from randomness, even when given $c = zG$ and $p_a G$ (which may be guessed by $\mathcal{P}_i$). The same argument applies to b and d. Since z is random, $c = zG$ is statistically indistinguishable from randomness.
- In step 4, $\mathcal{P}_i$ receives $\alpha = \hat{a}_s + \hat{a}_r$, which is indistinguishable from randomness since $\hat{a}_r$ is unknown to $\mathcal{P}_i$ given that the queried banks are not colluding. □

We note that one might also give a proof that proves that the protocol remains private when two of the three parties collude. This would require a more sophisticated simulator, which looks similar to that in the work by Vos et al. [58].

6.3 Security of the equality protocol

Finally, we prove the security of our secure equality protocol 3.

Claim 5. *Protocol 3 correctly and privately computes an equality.*

Proof. Verifying correctness of the secure equality protocol (Protocol 3) comes down to verifying its behavior depending on whether a party's coin tosses come out to an even or odd number of 1s. We study the case where there is one party, but the argument extends trivially to multiple parties.

- If $r_1 = 0$, then c_0 encrypts O and c_1 encrypts G. If c encrypts O, then $c' = c_0$. Otherwise, $c' = c_1$. Both are correct.
- If party $\mathcal{P}_i$ has $r_i = 1$, then c_0 encrypts G and c_1 encrypts O. If c encrypts O, then $c' = c_1$. Otherwise, $c' = c_0$. Both are correct.

Next, we analyze the security of Protocol 3. We do not consider the security of the ElGamal scheme, which we discussed previously. In the protocol, each party randomizes and shuffles the set of ciphertexts C. As a result, the only meaningful information that is revealed when the set is decrypted is the number of identity points. We refer to the number of non-identity points as t.

If a party would only perform random coin flips, the number of non-identity points t is given by $P(\boldsymbol{t} = t) = \binom{k}{t}0.5^k$. However, since we are inserting ciphertext c, this changes to $P(\boldsymbol{t} = t) = P(c = O)\binom{k-1}{t}0.5^{k-1} + P(c \neq O)\binom{k-1}{t-1}0.5^{k-1}$, where we use $c \neq O$ to denote that c does not encrypt O. Using this, we derive the posterior probability that $c \neq O$ given the number of points t.

$$\begin{aligned} P(c \neq O|\boldsymbol{t} = t) &= \frac{P(\boldsymbol{t} = t|c \neq O)P(c \neq O)}{P(\boldsymbol{t} = t)} , \\ &= \frac{\binom{k-1}{t-1}\cancel{0.5^{k-1}}P(c \neq O)}{(1-P(c\neq O))\binom{k-1}{t}\cancel{0.5^{k-1}}+P(c\neq O)\binom{k-1}{t-1}\cancel{0.5^{k-1}}}, \\ &= \frac{\binom{k-1}{t-1}P(c \neq O)}{(1-P(c \neq O))\binom{k-1}{t} + P(c \neq O)\binom{k-1}{t-1}} . \end{aligned}$$

The strongest attack is to guess $c = O$ when $P(c \neq O|\boldsymbol{t} = t) < \frac{1}{2}$ and $c \neq O$ otherwise. The expected guessing chance is then:

$$\sum_{t=0}^{k} P(\boldsymbol{t} = t) \underbrace{\left(\frac{1}{2} + \left|\frac{1}{2} - P(c \neq O|t = t)\right|\right)}_{\text{Guess based on the posterior}} . \tag{4}$$

An adversary who does not have access to the protocol's result can only guess using its knowledge about the prior probability of c, so it will succeed with probability $\frac{1}{2} + \left|\frac{1}{2} - P(c = O)\right|$. Using (4), we formulate the advantage of an adversary using our Protocol 3, and restrict it to 2^{-40}, which we deem negligible:

$$\left(\sum_{t=0}^{k} P(\boldsymbol{t}=t)\left(\frac{1}{2}+\left|\frac{1}{2}-P(x=1|t=t)\right|\right)\right)-\left(\frac{1}{2}+\left|\frac{1}{2}-P(c\neq O)\right|\right) \leq 2^{-40}. \tag{5}$$

The choice of k then depends on the probability of anomalies occurring. Let us consider $P(c \neq O) \leq 0.05$ as an example. Then, the first k for which (5) holds is $k = 44$. □

7 PERFORMANCE ANALYSIS

We empirically evaluated the utility and scalability of our solution for the detection of anomalies among millions of financial transactions in a federated setup with a payment network system (the centralized entity) and partner banks (the data-augmenting clients). To this end, we implemented our solution in *Flower*, a well-known framework for federated learning [11]. *Flower* assumes a star topology in which all data holders, including our centralized entity $\mathcal{S}$ and the data-augmenting clients, are connected to an aggregator that can only perform simple tasks like averaging and message passing. Since it does not support client peer-to-peer communication, we route all communication between $\mathcal{S}$ and the data-augmenting clients through the aggregator. We note that this is an implementation detail rather than a requirement originating from our protocol. We implemented Protocol 2 in Rust (with a Python wrapper).[2]

7.1 Experimental setup

Data. We use synthetic data provided by SWIFT to participants in the 2023 PETs prize challenge.[3] The data consists of two parts:

(1) Transaction data held by $\mathcal{S}$ (payment network system): a dataset of financial transactions that are labeled as anomalous (positive) or not. This dataset is split into a train dataset with 2,990,349 negative and 3,521 positive instances, and a test dataset with 1,002,395 negative and 1,279 positive instances. Each transaction known to $\mathcal{S}$ has, apart from other information, details of the ordering and beneficiary accounts, and the financial institutions involved in making the transaction. These financial institutions are the centralized entity's partner banks which hold information about the ordering and beneficiary accounts.
(2) Bank account data held by the data-augmenting clients $\mathcal{P}_1, \ldots, \mathcal{P}_n$: information about bank accounts. To test scalability, we distribute this data among a varying number of clients.

Features for model training by $\mathcal{S}$. We train model $\mathcal{M}$ (see Sec. 4.2) on the following features that are held by $\mathcal{S}$:

- Unexpected currency. Most transactions in the train data have the same InstructedCurrency and SettlementCurrency. The few transactions that involve two different currencies are all anomalous. We include `SameCurrency` as a feature.
- Unusual timestamps. We found the Timestamp and the SettlementDate to be strong indicators of anomalous transactions in the training data. We encode this as a feature `InterimTime` which is the Timestamp subtracted from the SettlementDate.

Consistency checks for Boolean feature extraction by $\mathcal{S}$ and the banks. We assume that $\mathcal{S}$ has a list of unique IDs pertaining to $\mathcal{P}_1, \ldots, \mathcal{P}_n$. In our implementation, $\mathcal{S}$ receives this information from the clients in a setup phase. For a transaction $\boldsymbol{x}$ involving a valid sending bank $\mathcal{P}_s$ and receiving bank $\mathcal{P}_r$ (i.e., $\mathcal{S}$ holds their IDs), $\mathcal{S}$ and the banks engage in a cryptographic protocol to compute a Boolean feature $B(\boldsymbol{x})$ derived from:

- Information from sending bank $\mathcal{P}_s$ indicating...
  - whether the account ID of the ordering entity as listed in the transaction $\boldsymbol{x}$ is a valid ID known to the ordering bank.
  - whether the name of the ordering entity as listed in the transaction $\boldsymbol{x}$ is the same as the name known to the ordering bank.
  - whether the street address of the ordering entity as listed in $\boldsymbol{x}$ is the same as the street address known to the ordering bank.
  - whether the country/city/zip of the ordering entity as listed in $\boldsymbol{x}$ is the same as that known to the ordering bank.
  - whether the ordering bank has flagged the ordering entity's account for any reason (e.g. account closed, account frozen,...).
- Information from receiving bank $\mathcal{P}_r$ indicating the same as above, but for the account of the beneficiary entity.

The Boolean feature $B(\boldsymbol{x})$ is 0 if the account information appears correct and 1 if there is any indication of inconsistency or unusual account information in either the sender or the receiver account.

[2]The code can be found at: https://github.com/steveng9/PETsChallenge.
[3]https://www.drivendata.org/competitions/group/nist-federated-learning/

Table 2: Utility-privacy tradeoff of models augmented with the consistency feature extracted jointly by $\mathcal{S}$ and the banks, averaged over 5 runs. Higher ϵ implies less privacy. The AUPRC results are independent of the number of clients. Anomaly detection consistently improves the AUPRC, even when the base model was already performing well.

Model	$\epsilon = 0.5$	$\epsilon = 1.0$	$\epsilon = 5.0$	$\epsilon = \infty$
RF	0.648	0.645	0.669	0.963
RF_{EMQF}	0.716	0.727	0.743	0.979
LR	0.869	0.908	0.915	0.943
LR_{EMQF}	0.892	0.925	0.935	0.964

with DP ($\epsilon < \infty$):
– RF-DP: Fletcher et al. [25]
– LR-DP: DP-SGD [3]
without DP ($\epsilon = \infty$):
– RF, LR: sklearn [48]

During the setup phase, each data-augmenting client sets up a single OKVS and key pair using Protocol 1. This protocol generalizes to any type of data as its only requirement is that the data is hashable. In our implementation, we let bank $\mathcal{P}_i$ encode R_i in the OKVS, which contains the `["Account", "Name", "Street", "CountryCityZip"]` columns. The bank omits any flagged entries from R_i (that is, where `flag != "0"`). $\mathcal{S}$ performs a different setup, only generating a single key pair.

During inference, $\mathcal{S}$ must check whether the ordering bank's dataset contains a row for the ordering user, and whether the beneficiary bank contains a row for the beneficiary. It does so using one invocation of Protocol 2. The selected fields are `["OrderingAccount", "OrderingName", "OrderingStreet", "OrderingCountryCityZip"]` and `["BeneficiaryAccount", "BeneficiaryName", "BeneficiaryStreet", "BeneficiaryCountryCityZip"]`.

7.2 Utility-privacy tradeoffs

The utility results in Tab. 2 are obtained by fitting models on the train dataset and evaluating them on the test dataset in terms of AUPRC (area under the precision-recall curve). We compare two kinds of models: Random Forest (RF) and Logistic Regression (LR). The models in Tab. 2 are trained on the feature set consisting of `SameCurrency` and `InterimTime`. However, the LR uses a discretized version of the `InterimTime` feature, as we explain in more detail below. In Tab. 2, we provide predictions made by all the models themselves, as well as when they are augmented with the Boolean feature representing the consistency check with the data-augmenting clients, denoted <model>$_{okvs}$ (Sec. 4.3).

The models are trained with algorithms that provide DP guarantees, under varying privacy budgets ϵ, corresponding to the different columns in Tab. 2. For comparison, in the last column we include results for models trained with an infinite privacy budget, i.e. with no DP guarantee at all. These models are trained with sklearn [48], using default values for the hyperparameters, with the exception of the use of 20 trees and max_depth = 10 for RF.

For ease of reference, in the text below we denote the models that we trained with a DP algorithm by appending "-DP" to indicate that they are differentially private. To train the RF-DP models in Tab. 2 we used the diffprivlib library [35], while for LR-DP we used the implementation of DP-SGD in TensorFlow Privacy [2]. Similar to the non-private setting ("$\epsilon = \infty$") from the last column in Tab. 2, the RF-DP models are trained with 20 trees and max_depth = 10. The way in which trees are constructed in this RF-DP approach [25] is quite different from the standard RF algorithm in sklearn that we used in the non-private approach. While in the standard RF algorithm each node in each tree is selected by evaluating it against the data, in the RF-DP approach, intermediate nodes and threshold values for these nodes are generated at random, to limit the number of queries needed against the data and stretch the privacy budget further. While in the non-private setting we obtained our best results with RF, this was no longer the case with RF-DP because the `InterimTime` feature only really pays off for well chosen thresholds. As mentioned earlier, the RF algorithm in the non-private setting was able to find and pick up those thresholds, while the RF-DP approach with all its random guessing of thresholds was not. As a result, in the federated setting, the LR-DP (on DP-SGD [3]) approach took over in terms of better utility, and, as we observed, was most stable across different runs.

DP discretization of `InterimTime`. We observed the `InterimTime` to be crucial for identifying anomalous transactions. Based on our observations in the non-private setting (the last column in Tab. 2), non-DP RF yields high AUPRC because the underlying decision tree learning algorithm has a built-in technique to find good thresholds for dynamic discretization of the `InterimTime` feature during tree construction. The DP training algorithms cannot detect such thresholds with the same ease. To mitigate this, we statically discretize the `InterimTime` feature into bins. We replace the `InterimTime` feature in each transaction with its corresponding bin number and one-hot-encode the bin numbers in the training set. In Appendix A, we explain how to achieve this while satisfying differential privacy.

Utility of consistency checks. As demonstrated in Tab. 2, there is a clear boost in accuracy when augmenting the centralized entity's model with the EMQF-based feature extraction protocol. The lower the accuracy of the model trained by $\mathcal{S}$, the greater the benefit of the data-augmenting feature extraction. Predicting anomalous instances solely from the feature extraction protocol (i.e. without training on the centralized entity's data) yielded an AUPRC of .294.

7.3 Efficiency and scalability

We performed efficiency and scalability experiments on a desktop Intel i7 6700k at 4.2GHz, 64GB memory, and GTX1080 GPU. The results in Tab. 3 are based on training a model, and running the consistency checks using three different client partitioning scenarios. In these scenarios, the average number of accounts per client is 561,935; 280,968; and 124,874; respectively. The runtimes, memory usages, and communication costs for $\mathcal{S}$ and clients change with the partition. Included in these metrics is the federated set-up, done in a privacy-preserving manner with the protocols from Sec. 4.3.

It can be seen in Tab. 3 that as the number of clients increases, the efficiency of the $\mathcal{S}$ decreases. This is expected since $\mathcal{S}$'s computation cost is largely dependent on the number of sender-receiver client pairs, which grows superlinearly with the amount of clients. However, the results also show that the total client runtime and memory resources remain fairly constant, and are mostly a function of the amount of client data rather than number of partitions. This can be explained by the fact that each clients' computational load is proportional to its data size. The total clients' communication cost

Table 3: Efficiency and scalability results on development data. The efficiency of the clients stays consistent as the partitions change, while $\mathcal{S}$'s efficiency increases. The client communication cost grows superlinearly as communication depends on the number of client *pairs* in our experiments.

	Time			Memory		Communication	
	Total	$\mathcal{S}$	$\mathcal{P}_i$	$\mathcal{S}$	$\mathcal{P}_i$	$\mathcal{S}$	$\mathcal{P}_i$
$\mathcal{S}$ + 2 clients	1596s	1198s	228s	3.50GB	1.95GB	1052B	1584B
$\mathcal{S}$ + 4 clients	1581s	1173s	234s	3.92GB	2.01GB	1200B	3168B
$\mathcal{S}$ + 9 clients	2701s	2215s	243s	4.36GB	1.85GB	2236B	7128B

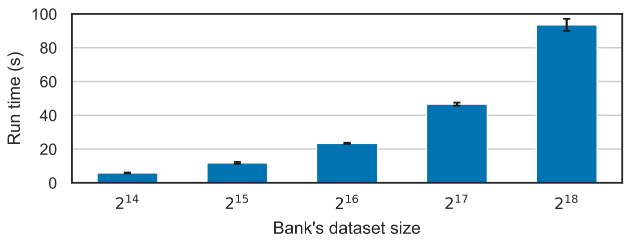

Figure 2: Time required for a bank to generate the OKVS using 8 threads on an M1 processor. Averaged over 10 runs, the error bars indicate the standard deviation. The run time scales linearly with the size of the bank's dataset.

on the other hand scales superlinearly with the amount of clients, since in our experiments, like in $\mathcal{S}$, this computation depends more on the amount of sender-receiver pairs.

Computational cost of the setup. We evaluated the run time of the setup by measuring how long it takes for a bank to create an OKVS. Since many parts of the PaXoS [50] OKVS algorithm can be parallelized, we ran this experiment on 8 threads. We present the results in Fig. 2. The run time scales linearly with the bank's dataset size. We note that while it takes approximately 90 seconds to generate an OKVS for 250,000 rows, this operation has to be computed only once (or whenever the dataset needs to change).

Cost of consistency checks. We also evaluated the run time of performing 128 consistency checks, split by the entities that take part in the protocol. We present the results in Fig. 3, where solid bars represent the measured computation time averaged over 10 runs, and transparent bars represent the time that a party would spend waiting to receive messages. This figure assumes that the latency is 100ms and we do not consider throughput constraints, given that all entities only exchange a small amount of compressed curve points which are made up of 32 bytes each. The run time for the centralized entity here scales linearly, while the data-augmenting entities perform a constant amount of work. Even at more than 250,000 rows, the centralized entity only spends 80 ms per query.

7.4 Comparison with other EMQFs

In Sec. 5.2 we discussed how there are multiple potential instantiations for a primitive that only returns encryptions of the identity when queried on elements in the set that it represents. Such a

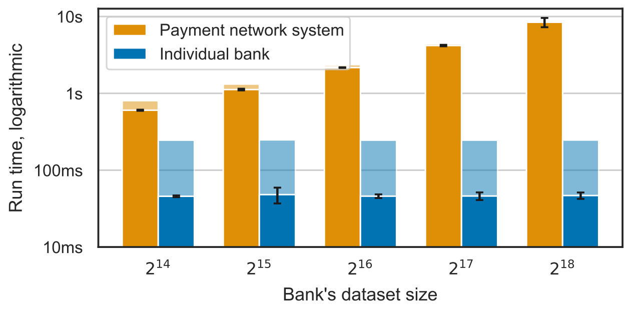

Figure 3: Time required for each party to complete 128 consistency checks on an M1 processor with one thread. Averaged over 10 runs, the error bars indicate the standard deviation. The run time is constant for the data-augmenting entities and scales linearly for the centralized entity. The transparent bars indicate time spent waiting for messages to arrive when the latency is 100ms, ignoring throughput constraints.

primitive allows moving a large fraction of computation and communication to a one-time (or each time that the set is updated) setup. We now compare the PaXoS OKVS with two alternatives that are used in MPSI protocols. Specifically, we compare PaXoS with the encrypted polynomials in the protocol by Hazay et al. [34] and the encrypted Bloom filters in the protocol by Vos et al. [58]. For all schemes, we use the same elliptic curve-based ElGamal ciphertexts. We measure the average time it takes to encode a set, to decode (query on an element), and the size of the OKVS or set representation. We present the results in Table 4 for a moderately-sized set of $k = 2^{14} = 16,384$ elements over 10 runs.

Notice how encrypted polynomials are the most compact but take the longest to encode. In fact, their size is optimal because the representation is exactly as large as the number of elements it contains multiplied by the size of a single ciphertext. Encrypted Bloom filters are significantly larger, even for large false positive probabilities μ. For the other schemes, the probability of a query returning a false positive is negligible. Finally, the PaXoS OKVS is efficient to encode and differs only a factor of 2.4 with the encrypted polynomial in terms of its size. This factor can be further reduced by increasing the number of hash functions [29]. One can also consider other OKVSs to instantiate our EMQF, e.g. trading off size and the time that it takes to encode or decode. Van Baarsen & Lu [57] provide another way of encoding curve points in an OKVS.

Table 4: Comparing EMQFs with $k = 2^{14}$ over 10 runs.

EMQF	Encode	Decode	Size
Encrypted polynomial	22.2s	1.03s	1 MB
Encrypted BF, $\mu = 0.1\%$	7.79s	0.02ms	18.8 MB
Encrypted BF, $\mu = 1\%$	5.19s	0.01ms	12.6 MB
PaXoS with ciphertexts	2.25s	0.03ms	2.4 MB

Table 5: Efficiency when performing p parallel queries

	$k = 2^{16}$			$k = 2^{18}$		
	Time	$\mathcal{P}_i$ sent	$\mathcal{S}$ sent	Time	$\mathcal{P}_i$ sent	$\mathcal{S}$ sent
Ours, $p = 1$	2.1 s	160 B	320 B	8.3 s	160 B	320 B
Ours, $p = 128$	2.7 s	20 kB	40 kB	8.9 s	20 kB	40 kB
Ours, $p = 512$	4.7 s	80 kB	160 kB	11.0 s	80 kB	160 kB
Kolesnikov et al.	79.6 s	956 MB	665 MB	327.2 s	3.84 GB	2.67 GB

7.5 Comparison with MPSI protocols

Standard MPSI protocols are not designed to be more efficient when some sets are fixed. We show that our work outperforms even fast MPSI protocols like the one by Kolesnikov et al. [42] by orders of magnitude. We use the popsicle library by Galois Inc. [27], which implements this protocol using the same elliptic curve library as in our implementation, assigning 1 thread per party. We report total run time, including setup time, and the bytes sent by $\mathcal{P}_i$ and $\mathcal{S}$ after setup. The implementation requires $\mathcal{S}$'s set to have the same size as the others' (same in the original), so the run time does not change whether the server queries $p = 1$ or $p = k$ elements in parallel. We report the cost per sequential query on an M1 processor.

Note that in [42], run time would be twice as low since each party has 2 threads. Their actual numbers are significantly lower, but even using their Table 3, the communication cost is 3 orders of magnitude higher than ours, while the run times are similar. Also, their protocol requires a full-mesh topology, incurring additional delays when communication is routed through a central entity.

8 LIMITATIONS

Protocol 2 discloses a single bit of private data from the data-augmenting entities to the central entity. This bit signals inconsistencies between data held by $\mathcal{S}$ and the data from data-augmenting entities $\mathcal{P}_i$. The implications of this single bit of information can be substantial depending on the nature of the data set held by $\mathcal{S}$. A zero bit immediately implies that the data entry held by $\mathcal{S}$ aligns with the data held by $\mathcal{P}_i$. In the case the leaked bit is one, there is also the potential for information leakage. within the specific context of the PETS prize, there exist multiple transactions with identical names (or addresses or banks). $\mathcal{S}$ can utilize the private set membership protocol results for these transactions with overlapping data to discern inconsistencies in these specific fields and recover them. For example, if two queries are made with the same name but the result of the query is different for both, then $\mathcal{S}$ will learn that the other fields are causing the discrepancy. Whether this release of information is acceptable depends on the particular application of our proposed protocols and needs a case-by-case analysis. The leakage of this information needs to be weighed against the societal cost of not detecting anomalous data.

The scenario described above is not unique to our proposed solution. It can occur in any situation where $\mathcal{S}$ trains a model with features sourced from $\mathcal{P}_i$ and where anomalies strongly correlate with data discrepancies between $\mathcal{S}$ and $\mathcal{P}_i$. In that case, learning the classification outcome of a transaction (anomalous or not) already provides significant insight regarding any potential data mismatch, exactly as in the case of our disclosed bit. For this reason, we loosely define the following requirements to decide whether our protocol can be applied in a given use case:

- There is a centralized entity with a global view of the system, but whose view can be enriched by incorporating data from data-augmenting entities.
- Data-augmenting entities only update their data at a low frequency, ensuring that OKVSs can be reused.
- All involved parties have an incentive to act semi-honestly. E.g., through legal obligations or financial incentives.
- The output of a private membership query may be revealed to the centralized entity. Or, when using Protocol 3, the output of the computation that follows it.

The release of information described above can be prevented by having the banks locally randomize their information and obtaining local differential privacy guarantees at the cost of reducing the utility of the final model. In the case the same query is repeated multiple times by $\mathcal{S}$, the privacy budget needs to be adjusted accordingly by using differential privacy's sequential composition property. This approach will typically not work, however, given that anomalies only make up a small proportion of the entire set of data. The reason is that the randomization will make the signal very noisy, significantly increasing the number of false positives.

Finally, we want to briefly state that while our protocol solves this specific privacy aspect, in practice one must take into account the wider context in which the protocol is deployed. One should be cautious about applying automated anomaly detection in general, as there are issues beside potential privacy violations that may negatively impact users. Nevertheless, our protocol fills an important gap in situations where anomaly detection is applied in federations.

9 CONCLUSION

Motivated by the PETS prize challenge, we propose an efficient solution for federated anomaly detection. Unlike traditional federated learning scenarios, our solution works for a case where the data is horizontally and vertically partitioned. Moreover, our solution is based on an efficient private feature extraction protocol - where features used in the training of a machine learning model are computed based on information distributed across different parties. Our proposed framework has applications beyond the specific scenario presented in the PETS competition. It proves valuable in any situation where inconsistencies across distributed data sets serve as important information for anomaly detection and does so while preserving privacy.

Despite the extensive literature on privacy-preserving machine learning focusing on protocols for private training of machine learning models over distributed datasets, our approach addresses a commonly neglected issue: privacy-preserving feature extraction. We privately compute features calculated over the distributed data set and subsequently use these features for training ML models. To accomplish this, we introduce an innovative private set membership protocol, combining the efficiency of oblivious key-value stores with inputs encrypted using elliptic curve-based ElGamal. By combining these two building blocks, the entities can perform membership queries with low computational overhead and bandwidth costs, while the only communication happens between the centralized entity and the involved data-augmenting entities.

ACKNOWLEDGMENTS

S. Golob and S. Pentyala are Carwein-Andrews Distinguished Fellows. S. Golob is supported by an NSF CSGrad4US fellowship. S. Pentyala is supported by the UW Global Innovation Fund and by a JP Morgan Chase PhD fellowship. M. De Cock and Z. Erkin thank the organizers of Dagstuhl seminar 22342 for jumpstarting the international collaboration that led to this paper. We would like to thank the reviewers and the shepherd for their helpful comments, and the organizers of the 2023 PETS prize challenge for the original motivation behind this work.

REFERENCES

[1] Aydin Abadi, Bradley Doyle, Francesco Gini, Kieron Guinamard, Sasi Kumar Murakonda, Jack Liddell, Paul Mellor, Steven J. Murdoch, Mohammad Naseri, Hector Page, George Theodorakopoulos, and Suzanne Weller. 2024. Starlit: Privacy-Preserving Federated Learning to Enhance Financial Fraud Detection. *IACR Cryptol. ePrint Arch.* (2024), 90. https://eprint.iacr.org/2024/090

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2021. TensorFlow Privacy. https://www.tensorflow.org/responsible_ai/privacy/guide.

[3] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 308–318. https://doi.org/10.1145/2976749.2978318

[4] Samuel Adams, Chaitali Choudhary, Martine De Cock, Rafael Dowsley, David Melanson, Anderson Nascimento, Davis Railsback, and Jianwei Shen. 2022. Privacy-preserving training of tree ensembles over continuous data. *Proceedings on Privacy Enhancing Technologies* 2 (2022), 205–226.

[5] Anisha Agarwal, Rafael Dowsley, Nicholas D. McKinney, Dongrui Wu, Chin-Teng Lin, Martine De Cock, and Anderson C. A. Nascimento. 2019. Protecting Privacy of Users in Brain-Computer Interface Applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27, 8 (2019), 1546–1555.

[6] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. 2017. More efficient oblivious transfer extensions. *Journal of Cryptology* 30, 3 (2017), 805–858.

[7] Hafiz Asif, Sitao Min, Xinyue Wang, and Jaideep Vaidya. 2023. *Anomaly Detection via Privacy-Enhanced Two-Step Federated Learning*. Technical Report. Rutgers University. https://rutgers.app.box.com/s/q84zjo3edv5d1e1eu67ypihiw8cb2djq

[8] Aslí Bay, Zekeriya Erkin, Jaap-Henk Hoepman, Simona Samardjiska, and Jelle Vos. 2022. Practical Multi-Party Private Set Intersection Protocols. *IEEE Trans. Inf. Forensics Secur.* 17 (2022), 1–15. https://doi.org/10.1109/TIFS.2021.3118879

[9] Daniel J. Bernstein. 2006. Curve25519: New Diffie-Hellman Speed Records. In *9th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2006) (Lecture Notes in Computer Science, Vol. 3958)*. Springer, 207–228. https://doi.org/10.1007/11745853_14

[10] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. 2013. Elligator: elliptic-curve points indistinguishable from uniform random strings. In *2013 ACM SIGSAC Conference on Computer and Communications Security*. 967–980. https://doi.org/10.1145/2508859.2516734

[11] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *CoRR* abs/2007.14390 (2020). arXiv:2007.14390 https://arxiv.org/abs/2007.14390

[12] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. 2021. When the curious abandon honesty: Federated learning is not private. *arXiv preprint arXiv:2112.02918* (2021).

[13] David Byrd and Antigoni Polychroniadou. 2020. Differentially private secure multi-party computation for federated learning in financial applications. In *ICAIF '20: The First ACM International Conference on AI in Finance, New York, NY, USA, October 15-16, 2020*, Tucker Balch (Ed.). ACM, 16:1–16:9. https://doi.org/10.1145/3383455.3422562

[14] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, Nadia Heninger and Patrick Traynor (Eds.). USENIX Association, 267–284. https://www.usenix.org/conference/usenixsecurity19/presentation/carlini

[15] Nishanth Chandran, Nishka Dasgupta, Divya Gupta, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Akash Shah. 2021. Efficient Linear Multiparty PSI and Extensions to Circuit/Quorum PSI. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). ACM, 1182–1204. https://doi.org/10.1145/3460120.3484591

[16] Melissa Chase, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, and Peter Rindal. 2017. Private collaborative neural network learning. *Cryptology ePrint Archive* (2017).

[17] Kamalika Chaudhuri and Claire Monteleoni. 2008. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou (Eds.). Curran Associates, Inc., 289–296. https://proceedings.neurips.cc/paper/2008/hash/8065d07da4a77621450aa84fee5656d9-Abstract.html

[18] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. 2011. Differentially Private Empirical Risk Minimization. *J. Mach. Learn. Res.* 12 (2011), 1069–1109. https://doi.org/10.5555/1953048.2021036

[19] Eduardo Chielle, Homer Gamil, and Michail Maniatakos. 2021. Real-time Private Membership Test using Homomorphic Encryption. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*. IEEE, 1282–1287. https://doi.org/10.23919/DATE51398.2021.9473968

[20] Yann Collet. 2021. xxHash – Extremely fast non-cryptographic hash algorithm. https://cyan4973.github.io/xxHash/.

[21] Ronald Cramer, Ivan Damgard, and Jesper Nielsen. 2015. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press Print, New York.

[22] Martine De Cock, Rafael Dowsley, Anderson C. A. Nascimento, Davis Railsback, Jianwei Shen, and Ariel Todoki. 2021. High performance logistic regression for privacy-preserving genome analysis. *BMC Medical Genomics* 14(23) (2021).

[23] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[24] Ahmed Roushdy Elkordy, Jiang Zhang, Yahya H Ezzeldin, Konstantinos Psounis, and Salman Avestimehr. 2022. How Much Privacy Does Federated Learning with Secure Aggregation Guarantee? *arXiv preprint arXiv:2208.02304* (2022).

[25] Sam Fletcher and Md Zahidul Islam. 2017. Differentially Private Random Decision Forests using Smooth Sensitivity. *Expert Systems with Applications* 78 (2017), 16–31.

[26] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1322–1333.

[27] Galois, Inc. 2019. swanky: A suite of rust libraries for secure computation. https://github.com/GaloisInc/swanky.

[28] Sanjam Garg, Mohammad Hajiabadi, Abhishek Jain, Zhengzhong Jin, Omkant Pandey, and Sina Shiehian. 2023. Credibility in Private Set Membership. In *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 13941)*, Alexandra Boldyreva and Vladimir Kolesnikov (Eds.). Springer, 159–189. https://doi.org/10.1007/978-3-031-31371-4_6

[29] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2021. Oblivious Key-Value Stores and Amplification for Private Set Intersection. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12826)*, Tal Malkin and Chris Peikert (Eds.). Springer, 395–425. https://doi.org/10.1007/978-3-030-84245-1_14

[30] Satrajit Ghosh and Tobias Nilges. 2019. An Algebraic Approach to Maliciously Secure Private Set Intersection. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 11478)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, 154–185. https://doi.org/10.1007/978-3-030-17659-4_6

[31] S. Dov Gordon, Carmit Hazay, and Phi Hung Le. 2022. Fully Secure PSI via MPC-in-the-Head. *Proc. Priv. Enhancing Technol.* 2022, 3 (2022), 291–313. https://doi.org/10.56553/POPETS-2022-0073

[32] Xiaolan Gu, Ming Li, and Li Xiong. 2021. PRECAD: Privacy-Preserving and Robust Federated Learning via Crypto-Aided Differential Privacy. *arXiv preprint arXiv:2110.11578* (2021).

[33] Chuan Guo, Awni Hannun, Brian Knott, Laurens van der Maaten, Mark Tygert, and Ruiyu Zhu. 2020. Secure multiparty computations in floating-point arithmetic. *arXiv:2001.03192* (2020).

[34] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. 2017. Scalable Multi-party Private Set-Intersection. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography,*

Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 10174), Serge Fehr (Ed.). Springer, 175–203. https://doi.org/10.1007/978-3-662-54365-8_8

[35] Naoise Holohan, Stefano Braghin, Pól Mac Aonghusa, and Killian Levacher. 2019. Diffprivlib: the IBM differential privacy library. *ArXiv e-prints* 1907.02444 [cs.CR] (July 2019).

[36] Roi Inbar, Eran Omri, and Benny Pinkas. 2018. Efficient Scalable Multiparty Private Set-Intersection via Garbled Bloom Filters. In *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 11035)*, Dario Catalano and Roberto De Prisco (Eds.). Springer, 235–252. https://doi.org/10.1007/978-3-319-98113-0_13

[37] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. 2018. Distributed learning without distress: privacy-preserving empirical risk minimization. In *Advances in Neural Information Processing Systems 31*. 6346–6357.

[38] Swanand Ravindra Kadhe, Heiko Ludwig, Nathalie Baracaldo, Alan King, Yi Zhou, Keith Houck, Ambrish Rawat, Mark Purcell, Naoise Holohan, Mikio Takeuchi, Ryo Kawahara, Nir Drucker, Hayim Shaul, Eyal Kushnir, and Omri Soceanu. 2023. Privacy-Preserving Federated Learning over Vertically and Horizontally Partitioned Data for Financial Anomaly Detection. *CoRR* abs/2310.19304 (2023). https://doi.org/10.48550/ARXIV.2310.19304 arXiv:2310.19304

[39] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[40] Alireza Kavousi, Javad Mohajeri, and Mahmoud Salmasizadeh. 2021. Efficient Scalable Multi-party Private Set Intersection Using Oblivious PRF. In *Security and Trust Management - 17th International Workshop, STM 2021, Darmstadt, Germany, October 8, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13075)*, Rodrigo Roman and Jianying Zhou (Eds.). Springer, 81–99. https://doi.org/10.1007/978-3-030-91859-0_5

[41] Marcel Keller and Ke Sun. 2022. Secure quantized training for deep learning. In *International Conference on Machine Learning*. 10912–10938.

[42] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. 2017. Practical Multi-party Private Set Intersection from Symmetric-Key Techniques. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1257–1272. https://doi.org/10.1145/3133956.3134065

[43] Yehuda Lindell. 2017. How to Simulate It - A Tutorial on the Simulation Proof Technique. In *Tutorials on the Foundations of Cryptography*, Yehuda Lindell (Ed.). Springer International Publishing, 277–346. https://doi.org/10.1007/978-3-319-57048-8_6

[44] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. 1273–1282.

[45] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy (SP)*. 19–38.

[46] Ofri Nevo, Ni Trieu, and Avishay Yanai. 2021. Simple, Fast Malicious Multiparty Private Set Intersection. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). ACM, 1151–1165. https://doi.org/10.1145/3460120.3484772

[47] Manas A Pathak, Shantanu Rane, and Bhiksha Raj. 2010. Multiparty Differential Privacy via Aggregation of Locally Trained Classifiers. In *Advances in Neural Information Processing Systems 23*. 1876–1884.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[49] Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson Nascimento, and Martine De Cock. 2022. Training Differentially Private Models with Secure Multiparty Computation. Cryptology ePrint Archive, Report 2022/146. https://ia.cr/2022/146.

[50] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2020. PSI from PaXoS: Fast, Malicious Private Set Intersection. In *39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12106)*. Springer, 739–767. https://doi.org/10.1007/978-3-030-45724-2_25

[51] Sara Ramezanian, Tommi Meskanen, Masoud Naderpour, Ville Junnila, and Valtteri Niemi. 2020. Private membership test protocol with low communication complexity. *Digit. Commun. Networks* 6, 3 (2020), 321–332. https://doi.org/10.1016/j.dcan.2019.05.002

[52] Adam Smith. 2011. Privacy-preserving statistical estimation with optimal convergence rates. In *Proceedings of the 43th Annual ACM symposium on Theory of Computing*. 813–822.

[53] Jinhyun So, Ramy E Ali, Basak Guler, Jiantao Jiao, and Salman Avestimehr. 2021. Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning. *arXiv preprint arXiv:2106.03328* (2021).

[54] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. 2017. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 587–601.

[55] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction APIs. In *25th USENIX Security Symposium*. 601–618.

[56] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. 1–11.

[57] Aron van Baarsen and Sihang Pu. 2024. Fuzzy Private Set Intersection with Large Hyperballs. Cryptology ePrint Archive, Paper 2024/330. https://eprint.iacr.org/2024/330 https://eprint.iacr.org/2024/330.

[58] Jelle Vos, Mauro Conti, and Zekeriya Erkin. 2022. Fast Multi-party Private Set Operations in the Star Topology from Secure ANDs and ORs. *IACR Cryptol. ePrint Arch.* (2022), 721. https://eprint.iacr.org/2022/721

[59] Jelle Vos, Mauro Conti, and Zekeriya Erkin. 2024. SoK: Collusion-resistant Multi-party Private Set Intersections in the Semi-honest Model. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA. https://doi.org/10.1109/SP54263.2024.00079

[60] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. SecureNN: 3-Party Secure Computation for Neural Network Training. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019), 26–49. https://doi.org/10.2478/popets-2019-0035

[61] Haobo Zhang, Junyuan Hong, Fan Dong, Steve Drew, Liangjie Xue, and Jiayu Zhou. 2023. A Privacy-Preserving Hybrid Federated Learning Framework for Financial Crime Detection. arXiv:2302.03654 [cs.LG]

A DIFFERENTIALLY-PRIVATE DISCRETIZATION OF THE `INTERIMTIME` FEATURE

To avoid any privacy leakage, we make the process of binning the `InterimTime` feature differentially private. To do so, we first compute the DP mean of the `InterimTime` feature for benign transactions.[4] A privacy budget of ϵ_1 is spent towards such computation. We then split the feature value range into two regions based on the above computed mean. Each region contains a distinct peak of benign samples. For each region, we compute the DP min and max value (percentile) of the `InterimTime` feature for the benign transactions and then generate 100 uniformly distributed bins for each percentile. The privacy of computation of percentile is due to [52] and a privacy budget of ϵ_2 is spent for computation of one percentile. Once the bins are generated for both the regions, these are one-hot-encoded. Each value of the feature `InterimTime` is then mapped to the corresponding one-hot-encoded bin number that it falls into.

The total privacy spent for computation of DP statistics for the binning process is $(\epsilon_1 + 2 \cdot \epsilon_2)$ which follows from sequential composition of DP. In order to keep the privacy budget equal to ϵ, we divide it between ϵ_1, ϵ_2, and ϵ_m, where ϵ_m is used in training the model with discretized bins. The total ϵ is allocated as $\epsilon_1 = \epsilon \cdot 1/50$, $\epsilon_2 = \epsilon \cdot 9/100$, and $\epsilon_m = \epsilon \cdot 4/5$, thus $\epsilon = \epsilon_1 + 2 \cdot \epsilon_2 + \epsilon_m$.

B CHANGES IN THE MALICIOUS MODEL

In the semi-honest model, the parties always follow the protocol specification. In the malicious model, a party is allowed to divert from this behavior. In the financial fraud detection application, one possible attack that is not addressed in our protocol might be for the data-augmenting entities involved in a transaction to collude with

[4]We use diffprivlib [35] to compute DP mean, min, and max, and we provide bounds for clipping that are independent of the data and depend on the given problem.

a user who makes a fraudulent transaction, covering their tracks by returning a fresh encryption of zero instead of randomizing the ciphertext sent by the centralized entity. One can prevent this attack by letting the data-augmenting entities provide a zero-knowledge proof that proves that the randomization is performed correctly. Note, however, that data-augmenting entities can also perform this attack by encoding the fraudulent data into the OKVS to begin with. This is not covered in the malicious model, as a party is free to choose their own inputs.

We may also assume that the centralized entity acts maliciously. For example, it could ask the data-augmenting entities to decrypt a different ciphertext than the one originating from the previous steps in the protocol execution. This too can be prevented using zero-knowledge proofs, where the centralized entity proves to the data-augmenting entities that the ciphertext they are asked to decrypt indeed corresponds to the randomized sum that they expect.

Beyond this, one can argue that the centralized entity should be limited in the number of queries that it makes: if it can perform arbitrarily many queries, then it can learn the data-augmenting entity's entire datasets. This can easily be done by data-augmenting entities, who can keep count of the number of protocol executions. Note, however, that this 'leakage' can also be seen as deliberate. For example, when it comes to the financial fraud detection application, the centralized entity must query a large number of transactions, potentially querying the same row multiple times. These all constitute valid transactions, and so the centralized entity is expected to learn whether these entities are in the datasets; the leakage is inherent to this use case.