Select-Then-Compute: Encrypted Label Selection and Analytics over Distributed Datasets using FHE

Nirajan Koirala*, Seunghun Paik†, Sam Martin*, Helena Berens*,
Tasha Januszewicz*, Jonathan Takeshita‡, Jae Hong Seo†, Taeho Jung*
*University of Notre Dame. Emails: smarti39@alumni.nd.edu; {nkoirala, hberens, ajanusze, tjung}@nd.edu
†Hanyang University. Emails: {whitesoonguh, jaehongseo}@hanyang.ac.kr
‡Old Dominion University. Email: jtakeshi@odu.edu

***Abstract*—Private Set Intersection (PSI) protocols allow a querier to determine whether an item exists in a dataset without revealing the query or exposing non-matching records. It has many applications in fraud detection, compliance monitoring, healthcare analytics, and secure collaboration across distributed data sources. In these cases, the results obtained through PSI can be sensitive and even require some kind of downstream computation on the associated data before the outcome is revealed to the querier, computation that may involve floating-point arithmetic, such as the inference of a machine learning model. Although many such protocols have been proposed, and some of them even enable secure queries over distributed encrypted sets, they fail to address the aforementioned real-world complexities.**

In this work, we present the first *encrypted label selection and analytics* protocol construction, which allows the querier to securely retrieve not just the results of intersections among identifiers but also the outcomes of downstream functions on the data/label associated with the intersected identifiers. To achieve this, we construct a novel protocol based on an approximate CKKS fully homomorphic encryption that supports efficient label retrieval and downstream computations over real-valued data. In addition, we introduce several techniques to handle identifiers in large domains, e.g., 64 or 128 bits, while ensuring high precision for accurate downstream computations.

Finally, we implement and benchmark our protocol, compare it against state-of-the-art methods, and perform evaluation over real-world fraud datasets, demonstrating its scalability and efficiency in large-scale use case scenarios. Our results show up to 1.4$\times$ to 6.8$\times$ speedup over prior approaches and select and analyze encrypted labels over real-world datasets in under 65 sec., making our protocol practical for real-world deployments.

I. Introduction

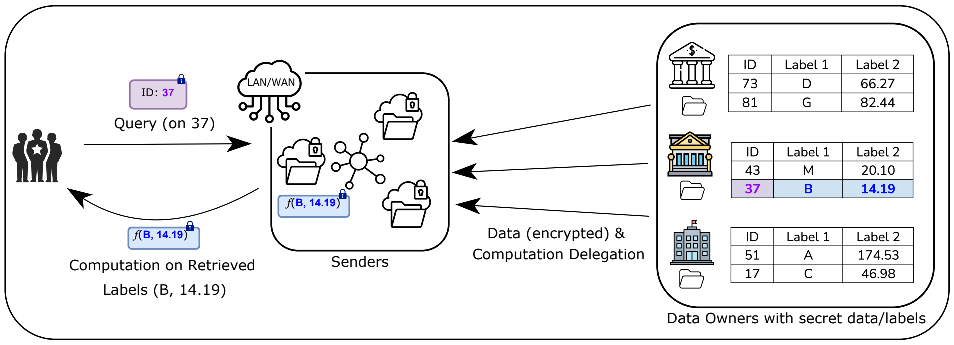
Modern data-driven workflows frequently necessitate collaboration among multiple independent data custodians. Such collaborations are subject to stringent privacy laws and rigorous security regulations. They are often only possible when both privacy and utility needs are met for all the involved parties. Across regulated domains, including finance, healthcare, and the public sector, organizations must balance the utility of collaborative analytics with statutory obligations to safeguard personally identifiable information (PII) [1], [2].

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240207
www.ndss-symposium.org

In the financial sector, institutions typically have limited visibility of their own customers' financial activities [3]. This fragmented view arises from the distribution of customer transactions across numerous entities, including other banks, credit card issuers, mortgage lenders, and mobile banking platforms. This fragmentation degrades fraud controls as models operate on partial views, inflating false positives and wasting analyst effort, while manual investigations often extend for weeks to months, delaying mitigation [4]. Collaboration that could close these gaps is impeded by onboarding friction, rising fraud pressure, and inter-institutional distrust, amplifying systemic risk. Privacy-preserving, cross-institutional analytics that assemble comprehensive risk profiles can reduce operational costs, mitigate financial risks, enhance profitability, and significantly accelerate investigations. Realizing this, however, demands strict compliance with financial and data-protection regulations [5]–[7] and credible mechanisms to resolve competitive sensitivities and institutional distrust. Similar challenges occur in healthcare, where sensitive electronic records, imaging, labs, and claims are siloed across hospitals, clinics, labs, and insurers; interoperability gaps and HIPAA constraints limit cross-institution visibility, impeding timely diagnosis, surveillance, and care coordination [8], [9]. In real estate, where brokers, lenders, insurers, and listing platforms each hold sensitive financial and personal data, regulatory obligations (e.g., GDPR/CCPA) and competitive sensitivities discourage direct sharing, exacerbating fragmentation [10].

Moreover, such sensitive datasets can be massive, which intensifies the complexity of ensuring utility without compromising confidentiality. At scale, custodians delegate storage and compute to the cloud [11] yet require designs that never expose plaintext to the provider. Regulations compel protection of customers' data [12], and providers likewise prefer encrypted delegation to limit breach liability [13]. In sum, the widespread fragmentation of massive sensitive data across institutions highlights the critical need for accurate and scalable cross-custodian analytics that preserve data confidentiality and comply with regulatory requirements.

In many scenarios, these datasets include both PII identifiers and sensitive PII labels (or payloads) associated with them, and the capability to privately compute over such labels can unlock critical use cases [14]. For example, within anti-money-laundering (AML) efforts that can span thousands of banks


ID: 37
Query (on 37)
LAN/WAN
f(B, 14.19)
f(B, 14.19)
Computation on Retrieved Labels (B, 14.19)
Senders
Data (encrypted) & Computation Delegation
Data Owners with secret data/labels

ID	Label 1	Label 2
73	D	66.27
81	G	82.44

ID	Label 1	Label 2
43	M	20.10
37	B	14.19

ID	Label 1	Label 2
51	A	174.53
17	C	46.98

Fig. 1: Overview of encrypted label selection and analytics (ELSA) over-distributed (data, label) pairs. f designates a computation function (e.g., logistic regression, ML model) over the labels of the matching data. Senders and receiver are connected using a LAN or WAN.

[15], institutions must securely identify PIIs such as account numbers and confidentially analyze associated financial data for risk scoring, all while preventing any disclosure of which institution provided the matching data [16]–[18]. Similarly, watchlist checks require agencies to securely query PII identifiers (passport numbers, national IDs) and privately retrieve and compute on related PII intelligence (e.g., arrest records, associates, travel histories) without revealing non-matching records [19]. Recent commercial offerings (e.g., Duality's Collaboration Hub with Oracle [20]) signal strong market demand for such encrypted analytics over fragmented financial datasets. In healthcare, providers or researchers require privately linking patient identifiers across hospitals, clinics, and laboratories and perform analytics on biomarkers (e.g., lab results) for disease-risk prediction and other diagnoses in compliance with interoperability rules [8], [21]. To achieve robust privacy-preserving analytics across large-scale, data-fragmented environments, the aforementioned real-world examples highlight the need for a privacy-preserving protocol that satisfies the following (high-level) requirements.

1) **Label confidentiality.** Sensitive label vectors (e.g., transaction histories, credit data) linked to identifiers must remain confidential end-to-end during selection and analytics.
2) **Privacy-preserving analytics.** Modern risk rating increasingly relies on statistical and ML models [22]; the system must support complex computations (e.g., inference, risk scoring) on PIIs such as associated labels to the identifiers without compromising privacy.
3) **Data fragmentation.** High-dimensional identifier–label records are distributed across many custodians under independent governance; secure federated selection and computation must proceed without centralizing data and scale with both the number of parties and data volume.

Querying large-scale, geographically distributed datasets is computationally intensive but feasible in non-private settings [23], [24]. Introducing strict privacy requirements, however, significantly increases both the computational and communication overheads. In the canonical workflow, a querier starts with a single unique identifier, say y (e.g., a bank-A/C number, passport ID, etc.) and must answer one compound question:

1) Selection: Test whether any custodian holds identifier y.
2) Label analytics: On a hit, select $\text{labels}(y)$ and compute $f(\text{labels}(y))$; otherwise, abort.

The two-stage pipeline decouples selection from computation and runs expensive private analytics only on positively matched records. Existing primitives—PSI, labeled PSI, circuit PSI, PIR, and their variants—fundamentally stop at the selection stage. They either reveal the intersection or nothing, support only restricted (often symmetric) function classes and data types, and cannot be directly lifted to encrypted analytics without leaking information or lack native support for real-valued payloads. Consequently, executing both stages—selection and label analytics—under the aforementioned requirements and with low computational and communication overhead lies beyond current privacy-preserving techniques. These requirements jointly give rise to a multi-institution setting in which many custodians hold encrypted (identifier, label) records and the goal is to securely compute, across parties, an arbitrary (potentially real-valued) function on the labels for a queried identifier, if present, while revealing nothing about non-matches or intermediate values and disclosing only the final output to an authorized recipient. We refer to this functionality as encrypted label selection and analytics (ELSA).

We illustrate the ELSA functionality to be realized in this work with an AML use case in Figure 1. An investigator encrypts a PII account identifier (e.g., ID 37) and requests a risk score computed over sensitive label vectors held across multiple institutions. Each institution delegates an encrypted (identifier, label) table to the cloud under a common FHE public key; label vectors encode AML features per the FFIEC BSA/AML manual [25] (e.g., transaction anomalies, regulatory reports). The protocol (i) privately selects the identifier across institutions, (ii) homomorphically computes the risk function on matched labels, and (iii) returns only the encrypted score, without revealing any raw labels.

Limitations of existing methods. There are several variants of PSI supporting label retrieval or computation over them. For instance, labeled-PSI [26], [27] allows the *receiver* to obtain associated data (labels) of matched items held by the *sender*. Circuit-based PSI [28]–[32] and its variants, e.g., private matching for compute (PMC) [33] or private join and compute (PJC) [34], [35], allow computations on the intersections.

However, existing protocols do not satisfy the requirements of ELSA. Labeled PSI necessarily reveals plaintext labels to the receiver and therefore cannot support computations over labels without disclosure. Circuit-based PSI can realize generic computations via SMPC, but requires multiple rounds of heavy interaction and does not natively support real-valued arithmetic; as a result, complex real-valued tasks (e.g., ML inference) incur substantial circuit and computational overhead. Moreover, these constructions typically assume plaintext storage of ID–label pairs and are designed for two-party settings, making extensions to many servers with encrypted data non-trivial. Recent work on PSI/PMC over encrypted data (e.g., Koirala et al. [36] and Mouris et al. [37]) scales to thousands of parties, but either does not support label retrieval and downstream analytics, or only enables restricted operations such as left joins.

Our Solution. In this work, we propose a one-step asynchronous protocol for achieving the ELSA functionality that computes on real-valued payloads with no extra receiver and sender interactions. Our protocol supports arbitrary computations on the (encrypted) labels resulting from matched records, which are distributed across a large number of senders, e.g., thousands or more. Existing methods fail in this setting either due to FHE-depth constraints, precision degradation, or prohibitive communication/compute costs, or do not support real-valued payloads or large identifier bit-lengths (δ). Our protocol offloads all heavy computations entirely to the resource-rich senders. It natively supports real-valued downstream computations, enabling rich analytics on labels derived from matched identifiers. Since these computations run under FHE across multiple custodians, key management is a major concern. For clarity, we initially assume a trusted third party (TTP) that provisions key management and distribution, and later remove this assumption by instantiating a threshold CKKS [38] scheme. Threshold-FHE is orthogonal to our design, and any type of threshold CKKS FHE is compatible with our protocol design. We introduce two major novel techniques: 1) generalization of the domain extension polynomial by Cheon *et al.* [39], and (2) slot-wise windowing technique. These two techniques enable our protocol to homomorphically evaluate the equality (or *if*) function over large domains with high accuracy and low FHE depth to be able to handle billions of identifier-label pairs for achieving ELSA. Based on these technical novelties, along with several other optimizations, we build a protocol for ELSA that is secure under a semi-honest adversary model. The contributions of this work are summarized as follows:

- We design the first CKKS-based protocol for ELSA that supports real-valued downstream computations without requiring SMPC or receiver-side heavy computations.
- We propose a novel approximation for homomorphic equality testing over large domains (2^{64} to 2^{128}), which can be efficiently evaluated while operating over small FHE presets.
- We propose several improvements and optimizations on the building blocks of our approximation method, including domain extension polynomials (Cheon *et al.* [39]) and slot-wise windowing, which sharply reduces the unit cost for the selection stage while ensuring high precision.
- We implement and evaluate our protocol on real-world datasets, demonstrating end-to-end evaluation in under 65 sec. We achieve speed-ups of up to 1.4× to 6.8× over prior state-of-the-arts in the selection setting. Our source code is available at https://doi.org/10.5281/zenodo.17849201.

II. Related Work

A. Private Set Intersection (PSI)

1) Circuit-PSI: Huang *et al.* [44] initially extended PSI to circuit-PSI, which enables parties with private input sets X and Y to privately compute any *symmetric* function over the intersection $X \cap Y$ via a boolean circuit [28], [45]. This computation allows symmetric functions such as cardinality, sum over associated attributes, or threshold intersection, which have many applications [46]–[49]. CHLR18 [50] extends FHE-based PSI to a labeled setting and describes how labels obtained after PSI can be masked and fed into a downstream generic SMPC. However, they mostly describe circuit-PSI over the labels in theoretical terms without any experimental validation. Additionally, the SMPC methods proposed in their work would induce a large communication overhead. Son *et al.* [29] built a circuit-PSI protocol based on the works of [26], [50], but their work only supports a few functions over the intersection and does not operate on labels.

Mahdavi *et al.* [27] proposed PEPSI, where the client sends an encrypted set, and the server computes an arbitrary function over the intersection. Their computation operates on the intersecting elements rather than their associated labels. Other works, such as [30]–[32], [51], [52], extend circuit-PSI to circuits on data associated with the intersection. However, the functions they compute are aggregations or cardinality, which are small circuits (nearly linear in the set size) to limit overheads. These constructions could be used for general post-intersection computation, but using them for more complex functions with floating-point operations can be very costly in terms of gate count and communication overhead.

2) Multi-Party PSI (MPSI): Kissner and Song [53] proposed one of the first works to yield intersection, union, and threshold functionality for MPSI, and subsequent works such as Nevo *et al.* [41] built maliciously secure MPSI protocols based on oblivious programmable PRFs (OPPRFs) and oblivious key-value stores (OKVS). Similarly, the O-Ring and K-Star protocols [42] are constructed from OPRFs and OKVS in the semi-honest model, supporting arbitrary collusions but, unlike our work, they do not provide encrypted-label analytics and operate over plaintext sets.

Chandran *et al.* [40] proposed multiparty circuit and quorum PSI protocols, which hide the intersection but evaluate a function on that intersection and identify which elements in a designated party's set appear in at least k other parties' sets.

Yang *et al.* [54] construct an unbalanced quorum MPSI protocol in a semi-honest, honest-majority setting using the BFV scheme; however, the server sets remain in plaintext (encoded as polynomials), and downstream computation is supported only via costly MPC circuits.

TABLE I: Comparison with related works. Notation: L.R and L.P denote Label Retrieval and Label Privacy, respectively. Real-Valued Func. denotes computational capability on input labels that are real numbers. − denotes not applicable. VAF and wDEP defined in Sections VI-A.

Protocol	Building Blocks	L.R	L.P	Computation over Label	Real-Valued Func.	Multi. Sender Scale	Security Model
Ion *et al.* [35]	OT + Hashing	✓	✓	Not supported	−	2-party	Semi-honest
Lepoint *et al.* [34]	OPRF + PIR + HE	✓	✓	Inner-product PJC only	−	2-party	Semi-honest
Cong *et al.* [26]	FHE + OPRF	✓	×	Supported via MPC*	×	2-party	Malicious
Rindal *et al.* [32]	OT (Vector-OLE) + Garbled circuit	✓	✓	Supported via MPC*	×	2-party	Semi-honest (opt. malicious)
Mahdavi *et al.* [27]	FHE	✓	✓	**Supported via FHE**	×	2-party	Semi-honest
Son *et al.* [29]	FHE + OPRF	✓	×	Supported via MPC*	×	2-party	Semi-honest
Mouris *et al.* [37]	Rerandomizable Encrypted OPRF	✓	✓†	Left-Join PMC only	−	1000s or more	Semi-honest
Koirala *et al.* [36]	Threshold FHE	×	−	−	−	1000s or more	Semi-honest
Chandran *et al.* [40]	OPPRF + MPC (garbled circuits)	×	−	−	−	Up to 15 parties	Semi-honest
Nevo *et al.* [41]	OPPRF + OKVS	×	−	−	−	Up to 32 parties	Malicious
Wu *et al.* [42]	OPRF + OKVS	×	−	−	−	Up to 100s	Semi-honest
Vos *et al.* [43]	EC-ElGamal + OKVS	×	−	−	−	Up to 100s	Semi-honest
This work	VAF, wDEP, Threshold FHE	✓	✓	**Supported via FHE**	✓	1000s or more	Semi-honest

*MPC requires a higher number of rounds of communication, as compared to FHE.
†Intersection sizes leaked to the delegator party (sender).

Other works like [55]–[59] have built (multi-party) private set union protocols, which is a problem setup different from ours.

3) Delegated PSI: In delegated PSI, each user, including the querying user, securely delegates their set to the cloud server, and ultimately, the querying user learns the intersection. Parties may not trust the cloud, which makes it similar to our setting. Kerschbaum *et al.* [60] proposed one of the earliest PSI protocols in which computation of the intersection is outsourced to an oblivious service provider. Duong *et al.* [61] presented PSI protocols to compute the cardinality of intersections for privacy-preserving contact tracing, and other works such as [62], [63] also operate under similar settings. Despite the efficiency and multi-party support in these works, there are still limitations for scenarios requiring long-term storage and queries over fully encrypted data. These works are specialized for only limited functions, similar to works under circuit-PSI, and their security crucially depends on no client colluding with the server or other clients, and they do not scale easily in terms of multiple delegated cloud servers.

B. Private Membership Test (PMT)

The private membership test (PMT) is a cryptographic primitive that allows a receiver to learn whether its singleton input y is contained in a large dataset X held by a sender. Recently, Garg *et al.* [64] have proposed a 2-round PMT protocol for string equality based on the DDH or LWE assumptions. [65] and [66] have also built PMT protocols based on the BFV and cuckoo filters, respectively. Other works, such as [58], [67]–[69], have also proposed various PMT protocols in the two or multi-party settings but operate over limited privacy settings for the datasets. Koirala *et al.* [36] recently proposed a threshold-FHE-based private segmented membership test (PSMT) protocol that aggregates responses from many servers in a single homomorphic summation. However, their work is limited to a yes/no membership result and does not retrieve any associated labels of the matching results from the server(s).

C. Other Similar Methods

Some works have focused on performing a downstream computation on the matched data after the match has been established privately. Ion *et al.* [35] proposed PSI with cardinality over the matched data for aggregating ad conversion rates. Lepoint *et al.* [34] defined the private join and compute (PJC) functionality that enables secure computation over data distributed across different databases. They expanded on [35] by enabling computation over the intersection but keeping the intersection itself hidden using Private Information Retrieval (PIR) and differential privacy techniques. Recently, Chida *et al.* [70] proposed a more communication- and round-efficient construction than [34]. However, these works are tailored towards computing a dot product of intersecting items and do not support arbitrary computations. Furthermore, they do not naturally extend to multi-party scenarios, and one would have to orchestrate multiple pairwise PJC computations or integrate them into an n-party SMPC [13].

Another line of work called private matching for compute (PMC) [33] enables aggregate downstream computation on the intersection. [37] and [71] improved upon the previous works by enabling private record linkage in a delegated setting and collaborative data analysis. One major limitation of these works is that they mainly focus on simple computations, like aggregation, and require additional SMPC for complex computations. Vos *et al.* [43] address repeated membership queries in cross-silo federations but reveal matching parties and only return a plaintext yes/no flag.

We compare representative prior works with ours in Table I.

III. Preliminaries

In this section, we summarize the key notations. y is the receiver's singleton query and sets X_i and ℓb_i are data owner's sets. δ denotes the length of items in y, X_i and ℓb_i. κ is the slot-wise windowing parameter. n, N, D, η denote the number of senders, FHE ring dimension, FHE depth, and batch size, respectively. $select_i$, $resLab_i$, $perm_i$, $flag_i$ and $c_{\ell b_{\text{stat}}}$ denote the output of the selection stage, ciphertext containing labels, ciphertext containing 1 in slot ρ_i, intersection indicator ciphertext (contains only 1 or 0) and statistics ciphertext respectively for the i^{th} sender such that $i \in \{1, \ldots, n\}$.

(Leveled) Fully Homomorphic Encryption: Fully Homomorphic Encryption (FHE) schemes allow computation on encrypted data. The Learning With Errors (LWE) problem and its variant Ring LWE (RLWE) underpin the security of modern FHE schemes [72], [73]. Since the first theoretical realization of FHE in 2009 [74], much work has been done

to make FHE more practically usable. The most prominent FHE schemes are BGV [75], B/FV [76], [77], CKKS [38], and TFHE [78]. For improved performance, the encryption parameters for FHE schemes are typically chosen to support only circuits of a certain bounded multiplicative depth (leveled FHE), which we use in our implementation. Our work uses the CKKS scheme as we deal with elements and associated data that are typically represented in floating-point arithmetic and can be used for analytics via ML, logistic regression, etc. Similar to B/FV and BGV, CKKS operates upon $\mathcal{R} = \mathbb{Z}[X]/\langle \Phi_M(X) \rangle$, where $\Phi_M(X)$ is the cyclotomic polynomial $(x^N + 1)$ of order $M = 2N$ (cyclotomic index) and degree $N \in \mathbb{Z}$ called the ring dimension. CKKS parameters include the ring dimension, ciphertext modulus q, scaling factor Δ, and the standard deviation of the error. In FHE schemes (except TFHE), SIMD (Single Instruction Multiple Data) encoding of plaintexts is possible, greatly increasing throughput. CKKS allows encrypting $\frac{N}{2}$ complex values into a single ciphertext.

Threshold FHE: For threshold functionality, we use an α-out-of-n threshold-FHE (thresFHE) scheme, where decryption requires α parties [79]. A thresFHE scheme consists of a tuple of probabilistic polynomial time (PPT) algorithms (*Enc*, *Eval*, *PartialDec*), and two n-party protocols (*KeyGen*, *Combine*) with the following functionalities:

- *KeyGen*$(1^\lambda, 1^D, params) \rightarrow (pk, evk, \{sk_i\}_{i\in[n]})$: Given a security parameter λ and a depth D, each party P_i outputs a common public key pk for encryption, a common evaluation key evk, and a secret key share sk_i of the implicitly defined secret key sk under some public parameter $params$.
- *Enc*$(pk, m) \rightarrow c$: Given a public key pk and a message m, the encryption algorithm outputs a ciphertext c.
- *Eval*$(evk, f, \{ck_i\}_{i\in[v]}) \rightarrow c^*$: Given an evaluation key evk, a v-input function f that can be evaluated using at most depth D and ciphertexts $\{c_i\}_{i=1}^{v}$, the evaluation algorithm outputs a new ciphertext c^* which is an encryption of $f(m_1, \dots, m_v)$, where $c_i \leftarrow$ *Enc*(pk ,m_i).
- *PartialDec*(c, sk_i): Given a ciphertext c and a secret key share sk_i, the partial decryption algorithm outputs a parital decryption result $pdec_i$.
- *Combine*$(pk, \{pdec_i\}_{i\in[I]}) \rightarrow m$ or $\perp$: Given a public key pk and a set of partial decryptions $\{pdec_i\}_{i\in[I]}$ for an index set $I \subseteq [n]$, the combine algorithm returns the decryption result m if $|I| \geq \alpha$ otherwise $\perp$.

During the partial decryption, a smudging noise from some predefined distribution is added to hide the secret key shares and the RLWE error accumulated from homomorphic operations [79], [80]. The key generation in thresFHE can be accomplished either using a trusted setup procedure to distribute partial secret keys, which can be executed via methods such as trusted hardware, or SMPC [80], [81]. We assume that the decryption threshold $\alpha < n/2$, i.e., honest majority setting. We require that a thresFHE scheme satisfy standard notions of compactness, correctness, and security [82].

Value Annihilating Functions (VAF): We use value annihilating function (VAFs) [36] as a building block for doing label selection and retrieval. It was defined by Koirala et al. [36] and informally, for a bounded domain $[-M, M]$, we say that the function f is a VAF if it satisfies $f(0) = 1$ and $f(x) \approx 0$ otherwise. From this property, we can observe that for $y \in [-M, M]$ and $X \subset [-M, M]$, $y \in X$ if and only if $\sum_{x\in X} f(y - x) > 0$, i.e., checking whether the additive aggregation of VAF evaluation results is zero or not.

IV. Encrypted Label Selection and Analytics

In this section, we introduce our main functionality, encrypted label selection and analytics (ELSA), which performs the following: (1) selects the encrypted labels over encrypted and distributed identifier-label pairs, (2) processes downstream computations over associated encrypted labels of the identifier.

Problem Formulation: We consider three types of entities: *data owner*, *sender*, and *receiver*. The data owner handles a dataset that consists of pairs $(x_i, \ell b_i) \in \mathcal{I} \times \mathcal{L}$, where $\mathcal{I}$ and $\mathcal{L}$ denote the universe sets for the identifiers and labels, respectively. We assume the data owner who, either due to resource constraints or operational preferences, delegates the encrypted database to a designated sender rather than participating directly. This setup parallels delegated PSI, freeing the owner from storing data locally or staying online [83].

Each sender is assumed to be a cloud server with strong computational and storage capabilities. They hold a pair of ciphertexts, one encrypting the unique identifiers and the other encrypting the corresponding labels. Following standard assumptions [84], [85], we assume the ordering of identifiers matches the ordering of labels. We designate one of the senders as the *leader sender* at the setup phase. The leader sender is responsible for evaluating a pre-defined function f, which takes encrypted labels held by each sender as inputs.

The receiver has access to a unique identifier as a query and learns the evaluation result of f on the labels of the matched elements. If a sender does not hold a matched identifier, it can result in the corresponding label becoming a "null" denoted as $\perp$, and the sender's labels would not contribute to the function computation. For this reason, we let the receiver learn separately whether the query is in the sender's set or not.

Function Evaluation: Our definition covers several types of functions analogous to existing circuit-PSI protocols. For instance, we can implement PSI with cardinality (f_{CA}) [31], which returns the number of matched items, or quorum PSI ($f_{\text{QR},\tau}$) [40] with a threshold τ, which returns a predicate bit indicating whether the number of matched items surpasses τ or not. If we set $\mathcal{L} = \{0, 1\}$, $\ell b_i = 1$ for all identifiers in $\bigcup_{i=1}^{l-1} \mathcal{X}_i$, and $\perp = 0$, then the following circuits implements each functionality, respectively.

$$f_{\text{CA}}(L) = \sum_{i=1}^{l-1} \ell b_i, \quad f_{\text{QR},\tau}(L) = 1\left(\sum_{i=1}^{l-1} \ell b_i > \tau\right)$$

where $L = (\ell b_1, \dots, \ell b_{l-1})$, which contains all the labels required to compute the respective function(s). Similarly, we can also implement several functionalities of PMC [33] or PJC [34], such as average or (weighted) summation on the associated labels.

Parameters: The protocol involves n parties: $\mathcal{S}_1, \ldots, \mathcal{S}_{n-1}$ senders and a single receiver $\mathcal{R}$. We denote the dataset outsourced to each S_i as $\mathcal{X}_i : i \in [n-1]$. Sets $\mathcal{I}$ and $\mathcal{L}$ contain the identifiers and corresponding labels, respectively. A function $f : \prod_{i=1}^{n-1}(\mathcal{L} \cup \{\perp\}) \to \mathbb{R}^m$ for $m \in \mathbb{N}$.
Inputs:

- $\mathcal{R}$: An identifier $y \in \mathcal{I}$.
- $\mathcal{S}_i$: An encryption of a set $\mathcal{X}_i \subset \mathcal{I} \times \mathcal{L}$.

Outputs:

- $\mathcal{R}$: Function evaluation result $f(\ell b_1, \cdots, \ell b_{l-1})$ and the flag bit $b \in \{0,1\}$, where ℓb_i is the associated data corresponding to $y = x_i$. If there is no matched identifier for the i'th sender, then $\ell b_i = \perp$. In addition, $b = 1$ indicates $y \in \bigcup_i \mathcal{X}_i$ and $b = 0$ otherwise.
- $\mathcal{S}_i$: Outputs $\perp$.

Fig. 2: Ideal functionality of ELSA

We emphasize that our notion of analytics on encrypted labels is much different from the standard circuit-PSI approaches that compute functions strictly on intersection sets. Traditional circuit-PSI works (e.g., [27]–[29], [31], [55]) focus on symmetric functions over intersected items, which do not align directly with our two-phase setting. In contrast, our approach enables computation on matched real-valued labels, i.e., $\mathcal{L} = \mathbb{R}$, reflecting practical use cases (see Section I). We state the ideal functionality for ELSA in Figure 2.

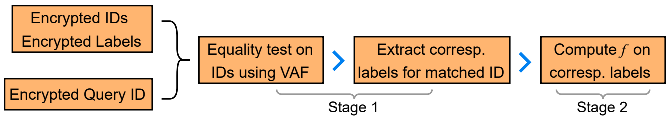
Security Model: We provide a security definition in the presence of a semi-honest setting where all parties act honestly but remain curious. Our protocol can be strengthened to achieve simulation-based security against a malicious receiver and preserve receiver-input privacy against a malicious sender using an OPRF pre-processing (CHLR18 [50]). Achieving full malicious-sender security, however, would additionally require ZKPs over large FHE circuits and is likely impractical in our setting. To realize threshold functionality in the full protocol, we use α-out-of-n thresFHE CKKS scheme where $\alpha < n/2$.

We assume adversaries, potentially corrupting up to $\alpha - 1$ parties, aim to learn either the receiver's query y (unless the receiver itself is compromised) or any sender's set element $x \in X_i$ and label $\ell \in \ell b_i$. To show the security in the presence of a semi-honest adversary, we need to show an efficient simulator that can simulate all the *views* of the corrupted parties, i.e., received communications during the protocol, through the inputs and outputs of the protocol. We provide the formal definition as follows:

Definition 1. *Let $\mathcal{F}$ be a n-party functionality and Π be a n-party protocol for parties $\mathcal{S}_1, \ldots \mathcal{S}_n$. We say that Π securely implements $\mathcal{F}$ at the presence of semi-honest adversaries with at most $(\alpha - 1)$ collusions if for every PPT adversary $\mathcal{A}$ controlling at most $(\alpha-1)$ colluding parties among $\mathcal{S}_1, \ldots \mathcal{S}_n$, there exists a simulator $\mathcal{P}$ such that for all PPT distinguishers $\mathcal{D}$: $|\Pr[\mathcal{D}(VIEW_{real}) = 1] - \Pr[\mathcal{D}(VIEW_{\mathcal{P}}) = 1] \leq negl(\lambda)$, where $VIEW_{real}$ is the view of the colluding parties during a real protocol execution and $VIEW_{\mathcal{P}}$ is the simulated view.*

V. Strawman Protocol and its Challenges

We instantiate our basic protocol as a strawman protocol. The core components of the strawman design are displayed in Figure 3. We temporarily assume a trusted third party



Encrypted IDs
Encrypted Labels
Encrypted Query ID
Equality test on IDs using VAF
Extract corresp. labels for matched ID
Compute f on corresp. labels
Stage 1
Stage 2

Fig. 3: Core components of ELSA protocol

(TTP) distributes the encryption, evaluation, and decryption keys to data owners, senders, and the receiver, respectively (this assumption will be removed in Section VII by introducing thresFHE). Each sender receives encrypted tuples of unique IDs and associated labels, $(c_{x_i}, c_{\ell b_i})$ from data owners via a secure channel. c_{x_i} is a encryption of $x_i \in \mathbb{Z}$ (or hashed δ-bit string), and $c_{\ell b_i}$ is its associated encrypted label. If a sender's set is larger than the batch capacity η, it stores multiple FHE ciphertexts. Senders are also equipped with a function $f(\{\ell b_i\}_{i \in L}; \Theta) = z$, where Θ can be a private value (e.g., ML model's private parameters [86]). The receiver holds a single query element $y \in \mathbb{R}$, which it replicates into all slots of the ciphertext c_y. The strawman protocol has two stages:

Stage 1 (Label Selection and Extraction): Upon receiving c_y from receiver, each sender S_i homomorphically,

1) Computes the difference of the query ciphertext c_y with its ciphertext c_{x_i} component-wise to obtain diff_i.
2) Applies a piecewise transform, value annihilating function (VAF) over diff_i that outputs 1 in each slot if $(y = x_i)$ for that slot, and 0 otherwise to obtain $select_i$.
3) Multiplies $select_i$ with $c_{\ell b_i}$ to obtain resLab_i. If multiple set encryptions exist, the sender sums all resulting resLab_i values into a single resLab_i.
4) Transmits its resLab_i to a designated leader sender.

Stage 2 (Computation on Labels and Result): At this stage, the leader sender receives the extracted labels from all other senders. The leader sums its own label ciphertext along with others, producing a combined label-collection ciphertext $\{\mathit{resLab}_i\}_{i \in L}$, where L is the set containing all the labels associated with the senders' element that matched query y. The leader then homomorphically evaluates the function $f(\mathit{resLab}_i; \Theta)$ to obtain $\mathbf{z}$, which is transmitted back to the receiver. The receiver decrypts $\mathbf{z}$ to obtain the final value (e.g., a numeric score), which can be a probability score or classification threshold.

Challenges for the Strawman Design. The strawman protocol provides a concrete realization of label selection and analytics under FHE. However, there are several significant challenges that must be addressed to ensure both correctness and efficiency in the strawman design. First, the core label-selection operation relies on a piecewise transformation (i.e., computing VAF), which homomorphically approximates the indicator function $1_{x=0}$ which outputs 1 if $x = 0$, and 0 otherwise. In the previous constructions [36], [39], [87], [88], coarse-grained precision sufficed during such approximation, which served to only distinguish intersection vs. non-intersection. Our setting requires a much higher-fidelity approximation of 1s and 0s: once an ID is matched with the query, the corresponding label must be recovered with a sufficiently small error so that further computations on these labels remain accurate. This approximation challenge becomes worst as the bit length of identifiers reaches 64 or 128 bits, significantly increasing the computational and communication overhead required to handle such large numeric domains. Next, beyond aggregating private label retrieval, our protocol evaluates a downstream function over the extracted sensitive labels. For entries not present in the intersection, the protocol must be able to exclude their label contributions altogether and signal the receiver to exclude the computation result for non-matched labels. We note that we focus on how to deal with labels resulting from non-matching items and not on ensuring the correct evaluation of the function itself. The verification of the computation of the correct function is an inherently hard problem in FHE and a separate line of research [89].

VI. Our Protocol

In this section, we present our techniques to solve the challenges associated with the strawman protocol. First, we motivate the label selection module of our protocol from existing works [36], [38], [66] and introduce a novel approximation for VAF that achieves provably higher accuracy with significantly lower computational and depth overhead (Section VI-A). Second, we propose slot-wise windowing, an efficient method for handling large set items (e.g., $\delta = 64$ or 128), which outperforms directly designing a VAF for δ-sized large domains (Section VI-B). This enables our protocol to scale to billions of identifier-label pairs with negligible false positives. Third, we present our method for label retrieval and optimized downstream computation on matched identifiers, achieving reduced memory overhead while revealing only the final result to the receiver (Section VI-C). Finally, we introduce a method to suppress labels from non-matching items, ensuring they do not affect downstream computations in Section VI-D.

A. Novel Approximation for the VAF

The VAF plays a crucial role in our protocol. In particular, it requires a VAF with indicator-level fidelity: it must evaluate to (near) one at exact equality and to negligibly small values elsewhere, so that slot-wise multiplication cleanly extracts labels and preserves accuracy for downstream computation. We present a novel VAF built from two independent components: (i) weak domain-extension polynomials (wDEPs) that compress wide input ranges without endpoint saturation and (ii) bell-shaped functions that concentrate mass at zero. We provide closed-form bounds on approximation error, multiplicative depth, and coefficient growth, enabling high-accuracy selection for 64–128-bit identifiers under CKKS.

Relation to prior VAFs: Prior VAF-based constructions for set-intersection (e.g., KTSJ24 [36]) target membership test and they approximate a tanh-derived curve via Chebyshev polynomials, then repeatedly square/scale the output and apply DEPs for large domains, reporting empirically tuned parameters. That line does not offer label extraction, encrypted real-valued analytics, or analytic guarantees needed for our pipeline. In contrast, our design is tailored to encrypted label selection and computation, with provable accuracy and complexity bounds and compatibility with the end-to-end CKKS setting.

1) Revisiting Domain Extension Polynomials: The DEP $D(x)$ shrinks the domain $[-L^n R, L^n R]$ to a smaller one $[-R, R]$ by behaving $D(x) = x$ when $x \in [-R, R]$, while $D(x) = R$ ($D(x) = -R$, resp.) for $x > R$ ($x < -R$, resp.). Then for a function f defined over $[-R, R]$, the composition $f \circ D$ becomes an extension of f to a larger domain $[-L^n R, L^n R]$ by regarding all the value outside of $[-R, R]$ as $f(R)$ or $f(-R)$ depending on the input's sign, while maintaining $(f \circ D)(x) = f(x)$ for $x \in [-R, R]$. However, when designing VAFs, we observed that it is not necessary for $D(x)$ to behave like the identity function near $[-R, R]$; in fact, the condition $D(x) = 0$ iff $x = 0$ is enough. Instead, to facilitate separating zero and nonzero inputs after applying the DEP, we focus on quantifying how far output values are from 0 for nonzero inputs. From these motivations, we propose a new definition of DEPs tailored for our purpose, called *weak DEPs* (wDEPs), as follows:

Definition 2. *For $0 < R_1 < R_2$, $0 < T < R_2$, and $\epsilon_T > 0$, we define $\mathcal{D}(R_1, R_2, T, \epsilon_T)$ as a class of polynomials $p : [-R_2, R_2] \to [-R_1, R_1]$ satisfying the following properties:*

- *$p(x) = 0$ if and only if $x = 0$.*
- *$\epsilon_T < |p(x)| < R_1$, $\forall x \in [-R_2, -T) \cup (T, R_2]$.*

We call $\frac{R_2}{R_1}$ a domain extension ratio. We say that the elements of $\mathcal{D}(R_1, R_2, T, \epsilon_T)$ is wDEP from $[-R_2, R_2]$ to $[-R_1, R_1]$.

The key difference between our wDEP and the original DEP [39] is that ours does not require the polynomial to behave like an identity mapping, e.g., conditions from [39] such as $|x - p(x)| < \omega |x|^3$ for some parameter $\omega > 0$ and $0 \le p'(x) \le 1$ for some small interval $(-r, r)$. Instead, we focus on quantifying the deviation of the function at the *tail* part of the domain, i.e., outside of $[-T, T]$.

Construction of wDEPs: We can construct wDEPs by the same domain extension method in the original DEP [39]; for some base function $D(x)$, it sequentially maps $x \mapsto L^i D(\frac{x}{L^i})$ for $i = n-1, n-2, \ldots, 0$. For the extension ratio $L > 1$, the following conditions are enough as a base function of wDEPs.

- $D(x) = 0$ if and only if $x = 0$.
- $|D(x)| \le 1$, $\forall x \in [-L, L]$.
- $\exists x_1, x_2 \in (-L, L)$ s.t. $D(x_1) = -1$ and $D(x_2) = 1$.

These conditions are closely tied to the domain extension process, so we first briefly explain how it works. Let $D(x)$ be a base function and suppose that we wish to extend it to $[-L^n, L^n]$[1]. Then for the input $y_n \in [-L^n, L^n]$, we recursively compute $y_i \leftarrow L^i D(\frac{y_{i+1}}{L^i})$ for $i = n-1$ to 0. We can observe that the input $\frac{y^{i+1}}{L^i}$ of $D(\cdot)$ always take a value between $[-L, L]$ if $D(z) \in [-1, 1]$ for any $z \in [-L, L]$. In addition, we also note that $D(\hat{z}) = \pm 1$ for some $\hat{z} \in (-L, L)$. If this is not the case, i.e., $\max_{x \in [-L, L]} D(x) = M < 1$, then $\frac{y^{i+1}}{L^i} \in (-M^{n-i+1} L, -M^{n-i+1} L)$. This implies that the output values of the resulting wDEP collapse to 0 for all inputs,

[1]For simplicity, we consider the case $R_1 = 1$.

so we cannot select desirable parameters (T, ϵ_T) for the tail parts. To sum up, if all these conditions above hold, then we can obtain a desired wDEP from the domain extension process.

For the choice of such a base function, we found that $D(x;k) = \frac{3\sqrt{3}}{2k\sqrt{k}}x(k-x^2)$ satisfies all these conditions. The first condition is straightforward as far as $x \in (-\sqrt{k}, \sqrt{k})$. To analyze the second and third ones, we can observe that the function takes extreme values at $x = \pm\sqrt{k/3}$, which can be obtained by a simple algebraic manipulation. In particular, $D(\sqrt{k/3};k) = 1$. Finally, $D(x;k)$ monotonically decreases at $x > \sqrt{k/3}$ and $D(\sqrt{k};k) = 0$. From this information, we can conclude that for $\max\{\sqrt{k/3}, 1\} < L < \sqrt{k}$, $D(x;k)$ satisfies all these conditions as a base function while belonging to $\mathcal{D}(1, L, \sqrt{k/3}, D(L;k))$.

Separation Factor: To quantify how much the given wDEP $f_{\mathrm{wDEP}} \in \mathcal{D}(R_1, R_2, T, \epsilon_T)$ separates values from 0, we define $\min\left\{|f_{\mathrm{wDEP}}(x)| : x \in [-R_2, R_2] \cap (\mathbb{Z} - \{0\})\right\}$ as a *separation factor*, i.e., the closest evaluation value to 0 for nonzero integer inputs. Since the image of f_{wDEP} becomes the domain of the subsequent function, and the input of wDEP is expected to be an integer, the set $\{f_{\mathrm{wDEP}}(x) : x \in [-R_2, R_2] \cap \mathbb{Z}\}$ determines the input domain of the subsequent function defined over the range $[-R_1, R_1]$.

The separation factor can be efficiently calculated by utilizing the properties of the wDEPs. We can observe that $|f_{\mathrm{wDEP}}(z)| \geq \epsilon_T$ whenever $|z| \geq T$. Hence, we can break down the range for measuring ϵ_{sep} by

$$\epsilon_{\mathrm{sep}} = \begin{cases} \epsilon_T & \text{if } T \leq 1, \\ \min\{\epsilon_T, \epsilon_{\mathrm{sep},T}\} & \text{otherwise} \end{cases}$$

where $\epsilon_{\mathrm{sep},T} = \min\{|f_{\mathrm{wDEP}}(x)| : x \in [-T, T] \cap (\mathbb{Z} - \{0\})\}$. Here, $\epsilon_{\mathrm{sep},T}$ can be found by $2\lfloor T \rfloor$ evaluations of $f_{\mathrm{wDEP}}(x)$. Moreover, if we use $f(x;k)$ the base function, then the resulting f_{wDEP} becomes a monotonic function for $[-T, T]$ if we select the smallest T for a fixed ϵ_T, i.e., $\epsilon_{\mathrm{sep},T} = f_{\mathrm{wDEP}}(1)$.

2) VAFs from Bell-Shaped Functions: For a wDEP $f_{\mathrm{wDEP}} \in \mathcal{D}(R_1, R_2, T, \epsilon_T)$ with a separation factor ϵ_{sep}, we can ensure that the inputs of VAF from nonzero inputs never fall into $(-\epsilon_{\mathrm{sep}}, \epsilon_{\mathrm{sep}})$. Hence, for a function f_{BS} defined over $[-R_1, R_1]$ such that the value outside the domain $(-\epsilon_{\mathrm{sep}}, \epsilon_{\mathrm{sep}})$ is sufficiently small to 0, we can observe that the composition $f_{\mathrm{BS}} \circ f_{\mathrm{wDEP}}$ behaves like the desired VAF. We formally define such a function as a (B, ϵ)-*bell-shaped function*, where the parameters ϵ and B correspond to ϵ_{sep} and the maximum evaluation value outside $(-\epsilon, \epsilon)$.

Definition 3. *Let $f : [-1, 1] \to [0, 1]$ be a function. For $B, \epsilon \in (0, 1)$, f is (B, ϵ)-bell-shaped if the following properties hold:*

- *$f(x) = f(-x)$, $f(0) = 1$, and $f(\epsilon) = B$.*
- *$\forall x \in [-1, \epsilon) \cup (\epsilon, 1]$, $0 \leq f(x) \leq B$, otherwise $f(x) \geq B$.*

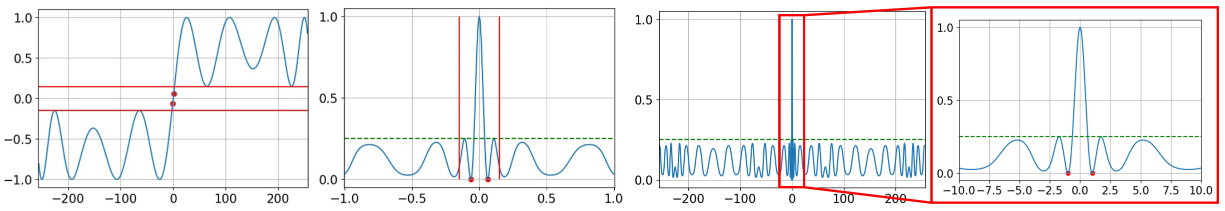
Transformation between Bell-shaped Functions: There would be several ways to design bell-shaped functions. Among them, we consider repeatedly applying the transformations between bell-shaped functions. For example, the squaring operation can be viewed as a transformation from (B, ϵ)-bell-shaped function $f(x)$ to another (B', ϵ')-bell-shaped function $g(x) = f(x)^2$ with $B' = B^2$ and $\epsilon' < \epsilon$. However, such a naïve squaring results in a vanishing of B before obtaining a sufficiently small ϵ. If $B = 0.5$ and we apply 12 squaring operations, then the final B becomes $B^{2^n} = 2^{-4096}$, which is too small to handle in usual parameters in CKKS.

To address the above issue, we aim to find a transformation that ensures $B' = B$ while reducing ϵ' as much as possible. To this end, we consider an affine transformation before squaring: $g(x) = (a \cdot f(x) + b)^2$ for some parameters a, b such that $a \geq 1$ and $a + b = 1$. We can observe that the range of $a \cdot f(x) + b$ in $[-1, -\epsilon) \cup (\epsilon, 1]$ becomes $[b, a \cdot B + b]$. Thus, the bound parameter B' of g becomes $\max\{b^2, (a \cdot B + b)^2\}$.

To reduce ϵ' as much as possible, we first show that ϵ is strongly related to the second derivative of f. If we consider a Taylor approximation of f at $x = 0$, we obtain $f(x) = 1 + \frac{f''(0)}{2}x^2 + O(x^4)$. This gives an approximation of the solution of $f(x) = B$ as $\sqrt{\frac{1-B}{-f''(0)}}$. Hence, increasing $|f''(0)|$ results in reducing ϵ. On the other hand, with a simple calculation, we can observe that $g''(0) = 2a \cdot f''(0)$. This implies that we need to select the largest possible a while maintaining the same B. To this end, we can select $a = 1 + \sqrt{B}$ and $b = -\sqrt{B}$; in this case $\max\{(a \cdot B + b)^2, b^2\} = b^2 = B$. Note that this enforces $\sqrt{B} < 1$ due to the definition, hence $a < 2$.

Depth-Efficient Transformation from Lazy Division: From the above approach, one caveat is that computing $(a \cdot f(x) + b)^2$ from $f(x)$ consumes two multiplicative depths in CKKS. Since $a < 2$ due to the parameter selection, squaring it two times greatly increases the second derivative. However, when a and b are rational numbers, we can mitigate this issue by applying a lazy division on their denominators. More precisely, let us denote $a = 1 + \frac{d}{q}$, $b = -\frac{d}{q}$ for some $d, q \in \mathbb{N}$ and $d < q$, $g_0(x) = f(x)$, and $g_{n-1}(x) = (a \cdot g_{n-1}(x) + b)^2$ for $n \in \mathbb{N}$. Here, instead of sequentially computing $g_2(x) = (a(a \cdot g_0(x) + b)^2 + b)^2$ from previous functions, we can apply the division by the denominator in a lazy manner to exploit the fact that the scalar multiplication by an integer can be done without any depth consumption. As we can save one depth for each procedure except for the last one, we can compute $g_n(x)$ from $g_0(x)$ with a total depth of $(n+1)$. With this technique, our transformation can increase the second derivative much faster than squaring under the same depth consumption. More precisely, we can observe that $g_n''(x) = 2^n(1 + \frac{d}{q})^n f''(0)$, while squaring $(n+1)$ times increases the second derivative by a factor 2^{n+1}. Thus, as long as $(1 + \frac{d}{q})^n \geq 2$, we can expect that this method reduces ϵ much faster than squaring.

3) Putting Them All Together: By composing the wDEP and bell-shaped function with appropriate parameters, we finally construct the desired VAF over the given range. We can ensure that the nonzero evaluation results are, at most, the bound parameter of the bell-shaped function, which can be precisely obtained in our approach. We summarize and visualize our result in Theorem 1 and Figure 4, respectively.



(a) Weak DEP (f_{wDEP}) (b) Bell-Shaped Func. (f_{BS}) (c) Final VAF from Composition ($g = f_{\text{BS}} \circ f_{\text{wDEP}}$)

Fig. 4: Visualization of the proposed VAF construction. The red dots correspond to intermediate values for evaluating $g(1)$ and $g(-1)$, and the red solid lines and green dashed lines correspond to ϵ_T and B, respectively. We can observe that for all nonzero integers x in the domain, $g(x) \leq B$ holds.

Algorithm 1 SLOT_WISE_WINDOWING

```
Require: Bit length $\sigma \in \{64, 128\}$; windowing count $\kappa \in \mathbb{N}$; integer element $e$ either in $[0, 2^{\sigma} - 1]$ or in $[-2^{\sigma-1}, 2^{\sigma-1} - 1]$.
Ensure: An array Windows $= \{c_1, \ldots, c_\kappa\}$, where each $c_i$ is in the domain $[-2^{\lfloor \sigma/\kappa \rfloor - 1}, 2^{\lfloor \sigma/\kappa \rfloor - 1}]$.
1: $exponent \leftarrow \lfloor \sigma/\kappa \rfloor$
2: $domain \leftarrow 2^{exponent}$
3: Initialize array Windows of length $\kappa$
4: for $i \leftarrow 1$ to $\kappa$ do
5:     $c_i \leftarrow e \bmod domain$          // Extract lower exponent bits
6:     $e \leftarrow \lfloor e/domain \rfloor$  // Reduce $\sigma$-bit integer by exponent bits
7:     if $c_i > \frac{domain}{2}$ then
8:         $c_i \leftarrow c_i - domain$        // Optional: map to negative range
9:     end if
10:    Windows$[i] \leftarrow c_i$
11: end for
12: return Windows
```

Theorem 1. *Let $f_{wDEP} : [-M, M] \to [-1, 1]$ be a weak DEP with a separation factor of ϵ_{sep} and $f_{BS} : [-1, 1] \to [0, 1]$ be a $(B, \epsilon_{\text{BS}})$-bell-shaped function. If $\epsilon_{\text{BS}} \leq \epsilon_{\text{sep}}$, then the composition $g = f_{BS} \circ f_{wDEP}$ of them has the following property: $g(0) = 1$ and $|g(x)| \leq B$ for any $x \in [-M, M] \cap (\mathbb{Z} - \{0\})$.*

We will use the following components to implement the proposed VAF. First, for the wDEP, we apply the domain extension technique for the base function $D(x;k) = \frac{3\sqrt{3}}{2k\sqrt{k}}(k - x^2)$. For the bell-shaped function, we apply our transformation $f(x) \mapsto (\frac{3}{2}f(x) - \frac{1}{2})^2$ several times on the function $f(x) = (1 - \frac{3}{2}x^2)^2$, which is a $(0.25, \frac{1}{\sqrt{3}})$ bell-shaped function. We additionally square the result five successive times to ensure that the final bound parameter is reduced to 2^{-64}. Parameter choices and presets are detailed in Appendix A, and efficient CKKS evaluation algorithms for wDEP and bell-shaped functions are given in Appendix A-B.

B. Supporting Long Items via Slot-wise Windowing

To achieve negligible false positives when encoding identifiers, such as when using hash-based representations, each identifier must be encoded with a sufficiently large bit-length to minimize collisions. For instance, using δ =64 or 128-bit encoding provides a robust guarantee against such collisions for a very large number of items. In previous PSI constructions based on finite-field FHE [50], [90], they encountered similar performance barriers for longer items (i.e., large bit-lengths δ). As δ grows, FHE parameters must be increased to avoid overflows and to preserve security, greatly increasing overhead. CHLR18 [50] addressed this issue by splitting large items into multiple smaller chunks and exploiting special SIMD encodings [91] that handle larger bit length items more efficiently. Even though we use CKKS, we face a similar problem during the approximation as approximating over large numeric ranges for high δ values is very costly, demanding substantial computation and FHE depth. Our novel VAF construction faces the same problem despite several improvements in approximation. As the depth of VAF evaluations significantly impacts the concrete efficiency, both in terms of communication and computation costs, it is necessary to optimize this core and computationally intensive component.

To address this issue, we propose a windowing technique for long items called *slot-wise windowing*. This method enables us to design a VAF for large domains using those defined over much smaller domains. For a precise description, let us assume that the sender's set elements (identifiers) are of some bit length, say δ-bits. Our key strategy is to parse the items into κ parts, so that each segment is represented by at most $\xi := \lceil \frac{\delta}{\kappa} \rceil$ bits. That is, we divide each identifier x into κ components $x_1, \ldots, x_\kappa$. Note that $x = 0$ if and only if $x_1 = \cdots = x_\kappa = 0$. Now, let $f_{\text{VAF},\xi}$ be a VAF defined for ξ-bit inputs. Then we can rewrite the previous condition by $x = 0$ if and only if $f_{\text{VAF},\xi}(x_i) = 1$ for all $i = 1, \ldots, \kappa$. This motivates us to build a function $f_{\text{VAF}}(x) := \prod_{i=1}^{\kappa} f_{\text{VAF},\xi}(x_i)$. We can immediately check that this function is indeed a VAF over the domain $[-2^\delta, 2^\delta]$. The choice of κ results in different communication-computation trade-offs in our protocol. We provide the algorithm for this technique in Algorithm 1.

We can observe that the above f_{VAF} is more depth-efficient than one directly built through techniques in Section VI-A, even in smaller domains. This is because κ evaluations of $f_{\text{VAF},\xi}$ can be computed in parallel, and the final multiplication only takes $\log_2 \kappa$ depths. We note that more slots are required to represent each set item as κ grows, which reduces throughput. In Section VIII-B3, we show that with a careful choice of κ, our windowing method provides robust throughput even when dealing with smaller domains.

C. Label Retrieval and Downstream Computation

We first introduce a method to optimize the downstream computation phase after the VAF-based label-retrieval on matched identifiers. Next, we address the key challenge from Section V by eliminating labels from non-matching items.

As detailed in Section V, our VAF enables each sender to efficiently extract labels by producing an *indicator* (1 for matched ID) that extracts the label from a target slot of label ciphertext via multiplication. The ciphertexts containing the labels after the extraction stage are then sent to one of the senders (leader). If there is a match, the label can reside in any one of the ciphertext slots; otherwise, all slots contain zeroes. To enable computations on these labels, the leader sender needs to be informed about the location of the slot that contains the extracted labels. Since the leader doesn't know which slot the label resides in, the leader can simply perform $log(\eta)$ rotations and additions to accumulate the values in all slots of $resLab_i$, such that each slot of $resLab_i$ contains the label (in case of a match) with some loss in precision which can be mitigated by an accurate VAF design. The leader can then operate on n-label ciphertexts, $resLab_i$, to compute any function homomorphically.

We observed that if the leader is able to aggregate multiple label ciphertexts into a single ciphertext, it can help reduce the memory overhead during computation. Aggregation enables the leader sender to operate on η slots of a single ciphertext instead of operating on n FHE ciphertexts, which can be costly in terms of memory. However, aggregating results label ciphertexts by the leader is not trivial: naïve additive aggregation fails due to unknown and overlapping slot positions. We resolve this with a technique we call *slot isolation*.

Optimizing Memory Overhead via Slot Isolation: First, each sender locally performs the $\log(\eta)$ slot rotations and additions to assemble $resLab_i$ before transmission, parallelizing the costly FHE operations and removing the leader as a computation bottleneck. Next, each sender multiplies *resLab*$_i$ by *oneHot*$_i$ plaintext vector, which has all zeroes except a single 1 in slot i, yielding a ciphertext whose label for i-th sender is isolated in slot i. If there are multiple types of labels (say $|\mathcal{L}|$), then *oneHot*$_i$ contains exactly $|\mathcal{L}|$ slots containing 1. The leader and other senders are informed about the number of multiple types of labels held by each sender at the setup phase. The leader homomorphically aggregates its own extracted label, along with all other *resLab*$_i$, to obtain $z = \sum_{i=1}^{n-1} resLab_i$, which contains the label for the i-th sender at the i-th or multiple $|\mathcal{L}|$ slots. Only one ciphertext z is then used during the function computation, instead of n such ciphertexts, reducing the memory overhead from $n-1$ FHE ciphertexts to one. This optimization requires $(n-1) < \eta$ when each sender contributes a single label type; if senders contribute multiple types, the number of distinct label types must instead satisfy $|\mathcal{L}| < \eta$. In practice, both n and $|\mathcal{L}|$ are typically well below η in most practical multi-party scenarios.

D. Dealing with Labels from Non-Match IDs

In our construction, the leader sender may receive *empty* ciphertexts (i.e., encryptions of the zero vector) when no intersection occurs. Since label values are unrestricted in our construction, a true label value of "0" may be indistinguishable from the near-zero output of a non-intersection during the function computation phase. On the other hand, when the distribution of labels is far from 0, providing "0" value to the function may result in an invalid computation outcome. Thus, appropriate mitigation is required to ensure correctness in the presence of empty ciphertexts.

We present two methods to preclude this issue. First, each sender designates one slot in common and embeds the intersection result into that slot, i.e., the summation of VAF results across all the data. Then, after aggregation, the resulting ciphertext contains the number of matches across all senders. The leader sender may apply another VAF to this number. With this idea, the receiver can decide whether the retrieved evaluation value was derived from actual matches or not. In addition, we also employ *imputation*, a common method in statistical analysis for dealing with missing variables [92]. In this technique, the missing variables are replaced by the statistics of the data, such as mean or median. In our scenario, these values can be pre-computed and delegated by the data owner. We observe that by viewing the evaluation result of the VAF as an *indicator*, each sender can decide whether to use the original label or the statistic value during label extraction. Hence, function evaluation becomes robust to missing values across senders, yielding reliable results.

VII. Discussion

a) Eliminating TTP via Threshold FHE: We remove the TTP assumption by instantiating our full protocol using an α-out-of-n CKKS thresFHE with noise flooding to achieve IND-CPA$^{\text{D}}$ security that tolerates up to $\alpha - 1$ colluding parties for $\alpha < n/2$. ThresFHE limits the full-exposure of the decryption key to a single party and removes the single point of failure by preventing any sub-threshold coalition from decrypting or attributing matches. It also strengthens data-owner trust in delegated storage as no single party, including the receiver, can recover plaintext ID–label pairs, and decryption is possible only to an authorized threshold coalition.

b) Duplicate Identifiers and Label Types: Our model assumes that duplicate IDs may exist across different senders' sets but not within a single sender's set. Data owners can simply preprocess the ID-label pairs before uploading them to the senders to achieve this, and this practice matches standard data-cleaning pipelines that remove duplicated data and noise before any privacy-preserving linkage [93]. Each sender packs its retrieved label into one CKKS ciphertext, and the leader evaluates f once per ciphertext, yielding $O(n)$ cost regardless of duplicate ownership. We permit each sender to include multiple types of labels in their dataset, but all senders share the same overall label schema (i.e., they hold labels of the same types). Our framework supports variable label cardinality per ID across different senders.

c) Provenance Security: Many works have discussed scenarios where the origin of an intersecting item must remain undisclosed, sometimes referring to it informally as "source anonymity" or "provenance privacy" [36], [94]–[97]. We can ensure provenance security in our protocol by permuting the slots in $resLab_i$ after the label retrieval, which hides the mapping between the senders and the slots. However, this makes the protocol inherently restricted to symmetric functions (summation, cardinality, etc.). Prior works [28] operate under this setting for computing over the intersection, thereby preserving item-level and provenance security. However, supporting asymmetric or general-purpose functions can offer significantly richer functionalities [14]. While asymmetric function computation may leak provenance, such leakage is limited to the leader. The receiver learns nothing unless it colludes with the leader. In practice, the leader is incentivized, both reputationally and economically, to preserve the provenance security of senders and ensure correct output. We provide a formal definition and proof for *provenance security*, which requires that adversarial parties cannot link items in the intersection back to the dataset owner(s), and the leakage is only limited to the party who is responsible for function computation. We provide the formal definition for such security in Definition 4 in Appendix B-B.

Our complete ELSA protocol is detailed in Figure 7 (Appendix A) after incorporating the aforementioned optimizations and mitigations. We prove the protocol secure in the semi-honest model against up to $(\alpha-1)$ colluding senders and the receiver, with a simulation-based proof in Appendix B.

VIII. Evaluation

A. *Experiment Setup*

We implement our protocol in C++17, using OpenFHE v1.2.3 [98]. The end-to-end protocol is instantiated with thresFHE CKKS and incorporates slot-wise windowing, slot isolation, and imputation for non-match IDs. Our tests were run on a single Intel Xeon(R) Gold 5412U server (512GB RAM, Ubuntu 22.04). Each experiment was repeated 10 times, and we report the average run-time. We used default thresFHE CKKS parameters in OpenFHE to maintain 128-bit classical security [99]. We assumed $\alpha = n/2$ for thresFHE and FHE ring dimension, $N = 2^{17}$. Unless otherwise specified, we treat each dataset item as a hashed value of length $\delta = 64$. We re-implemented all of the compared works, [26], [27], and [36], in OpenFHE under identical FHE parameter settings for a fair comparison. Each sender's computations are independent of each other in the multi-sender setting in our protocol. Once they receive the query ciphertext from the receiver, we assume each sender processes its own encrypted data in parallel. For noise smudging, we assume that senders perform a one-time, offline static noise estimation using publicly available input bounds based on real data and the evaluated VAF. This process depends solely on query computation and incurs negligible latency. In our LAN setting, we assume a 20 Gbps and 10 Gbps bandwidth with a 0.2 ms RTT, while WAN setup assumes 1 Gbps and 500 Mbps bandwidths with an 80 ms RTT.

Since there is no direct comparison for the same end-to-end functionality as ours, we compare only the selection stage against ELSA to state-of-the-art baselines. We evaluate the performance in terms of computational and communication costs and scalability in terms of the number of senders and set sizes for two variants of our protocol, (1) a selection stage-only benchmark and comparison against state-of-the-art prior works in Section VIII-B and Section VIII-B3, (2) a labeled version (i.e., ELSA with downstream computations) evaluation on three fraud-oriented datasets provided by Fraud Dataset Benchmark (FDB) [2] in Section VIII-C. These datasets were selected to evaluate the end-to-end performance of ELSA under high label and entry counts. The IEEE-CIS Fraud Detection dataset (IEEECIS) comprises around 590K card-not-present transactions with 25 labels. The Sparkov (Simulated) Credit Transactions dataset (CCTFD) includes 1.2 million entries with 24 labels, and the Vehicle Loan Default dataset (VLDP) consists of about 233K records with 44 labels. For the downstream computations, we employed logistic regression, a widely used model in applications such as fraud detection and disease prediction. We assume senders have access to a pre-trained logistic regression model $\sigma(\mathbf{w}^\top \mathbf{y} + b)$ which is parameterized by some private parameter Θ (e.g., a weight vector $\mathbf{w} \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$) in one instantiation.

TABLE II: Comparing methods to approximate VAF over a single sender with 2^{16}. $\kappa = 1$ denotes slot-wise windowing is disabled. Ctxt denotes FHE ciphertext.

Approx. Method	**Item Length (δ)**	**Sender Ctxt No.**	**Receiver Ctxt (MB)**	**FHE Depth**	**CKKS Precision (bits)**	**Time (s)**
KTSJ24	20	1	113.3	52	30.0	151.6
$\kappa = 1$	20	1	86.0	39	25.1	28.87
$\kappa = 2$	20	2	54.0	24	36.0	11.37
$\kappa = 4$	20	4	44.0	19	42.6	8.20
$\kappa = 5$	20	5	37.0	16	39.6	5.60
$\kappa = 10$	20	10	33.0	14	42.1	5.48

B. *Benchmarking the Selection Stage (Query Matching)*

In this section, we benchmark the performance metrics of the selection stage in ELSA, under varying numbers of senders and encrypted sender set sizes. We first discuss the preference criteria of the slot-wise windowing parameter κ. We then outline end-to-end latency in terms of both computational and communication, demonstrating our approach's scalability.

1) Evaluating Optimal κ for Supporting Large Items: Choosing optimal κ in the slot-wise windowing method for a particular sender set size is essential in our method. In Table II, we evaluate the impact of varying κ on communication, computation, and precision when approximating the VAF over a 2^{16}-sized sender set with 20-bit elements under a single-thread setting. The optimal value of κ in this setting can be determined to be either 5 or 10 (highlighted gray). κ= 5 reduces the sender storage while κ= 10 reduces the communication cost. Both achieve the lowest latencies while preserving high precision. For the evaluation of the datasets with a larger amount of records and δ value (as in FBD), we require different κ values when dealing with $\delta = 64$ or 128.

As a baseline, we compare our optimization method against a similar work (KTSJ24 [36]) that conducts CKKS-based approximation for membership testing.

2) Scalability in terms of Number of Senders and Set Sizes: The total latency (including communication cost) of

[2]https://github.com/amazon-science/fraud-dataset-benchmark

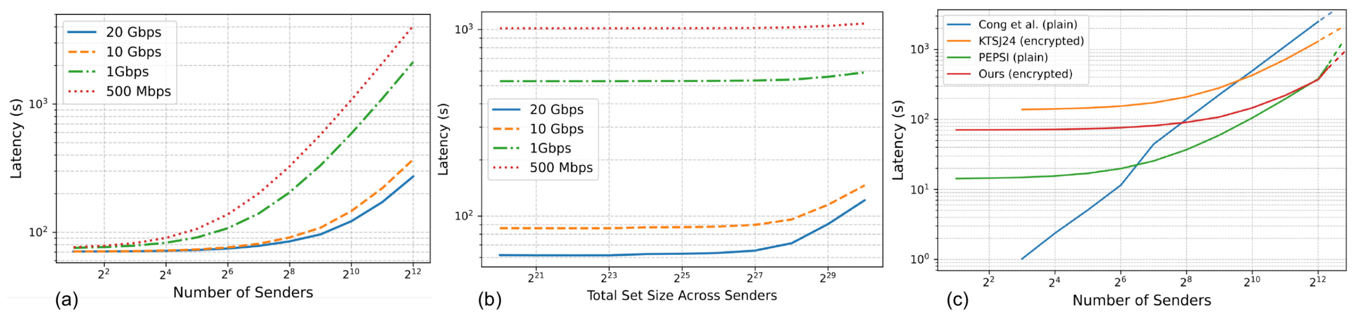


Fig. 5: (a) Latency of ELSA's selection stage for different number of senders (2^{20} items per sender), (b) Latency of ELSA's selection stage for different total sender set sizes across senders (1024 senders); $\delta = 64$ and $\kappa = 8$ in both (a) and (b). (c) Comparison of ELSA's selection stage with prior works. $\delta = 64$ for all except KTSJ24, $\kappa = 8$ for ours. Figures are drawn on a log scale.

our approach for an increasing number of senders under various network bandwidths is shown in Figure 5 (a). We can observe that our core selection stage easily scales for a high number of senders (up to 4096). For the applications discussed in Section I, the institutions are often equipped with high bandwidth LAN connections, and our total latency stays below 400 sec under such connections. We show the total end-to-end latency for various total sender set sizes in Figure 5 (b). Our method can easily support up to 2^{30} sender set size, and the overall latency remains below 150 sec under LAN settings.

3) Comparison with State-of-the-art: In this section, we compare our core functionality (query matching on encrypted records) against various state-of-the-art works. We evaluate the end-to-end latency (computation and communication) and show how it scales with the increasing number of senders and set sizes under a 10 Gbps LAN. Considering the existing works that we discussed in Section II, [40], and [41], are non-FHE-based works that support only a limited number of senders. We primarily compare our method against three state-of-the-art FHE-based works, Cong *et al.* [26], PEPSI [27], and KTSJ24. In PEPSI's client–server model, the client corresponds to our receiver and the server corresponds to our leader sender. PEPSI only encrypts the client's queries, so query matching is performed using ciphertext–plaintext operations. Note that both Cong *et al.* and PEPSI operate on plaintext sender sets in their protocols to enable various optimizations. Hence, we keep the same plaintext setting for them. Therefore, KTSJ24 and ours are disadvantaged in this comparison since they are run on encrypted sender sets (ciphertext-ciphertext operations). We use $\Delta = 59$ for CKKS-based methods (ours and KTSJ24). We set $\delta = 64$ for Cong *et al.*, PEPSI, and ours. For KTSJ24, we set $\delta = 25$ (the highest δ supported by their parameter presets), although it favors them. We note that at $\delta = 25$, ours is about 44$\times$ faster than KTSJ24. Ours, Cong *et al.*, and PEPSI achieve a negligible false positive rate (2^{-30} to 2^{-40}) from hash collisions when using 64-bit (δ) hashed items across all the sender set settings, while KTSJ24 exhibits non-negligible hash collisions. We note that ours can be easily extended to a higher δ, which will incur even smaller false positive rates ($< 2^{-100}$ when δ= 128). The end-to-end latency results are shown in Figure 5 (c) and Figure 6, where the maximum number of senders and set size is capped at 5000 and 2^{31}, respectively, due to memory constraints.

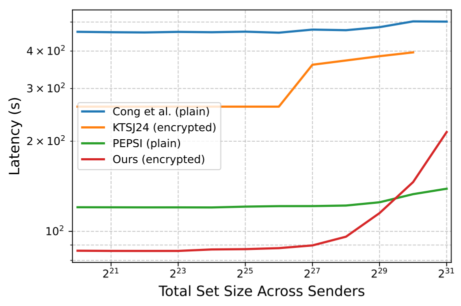


Fig. 6: Comparison of ELSA's selection stage with other works. The number of senders is set to 1024. $\kappa = 8$ for ours. $\delta = 64$ for all except KTSJ24. KTSJ24 omitted at 2^{31} (unsupported set size).

Our method achieves significantly lower latency ($3.5\times - 6.8\times$) compared to Cong *et al.* and KTS24 in Figure 5 (c). Compared to PEPSI, our protocol has higher latency when the number of senders is less than 4096, but we operate on fully encrypted datasets. However, ours gets faster (projected dotted lines in Figure 5 (c)) as we cross 4096 senders. In terms of varying sender set sizes shown in Figure 6, ours is faster ($1.4\times - 5.4\times$) than other methods. PEPSI starts to gain speed up over ours for larger set sizes since they operate over plaintext sender sets and cannot support downstream computations. As the sender set sizes exceed 2^{29}, ours must hold more ciphertexts in memory (e.g., 256 for 2^{31} entries) and evaluate VAF on each of them simultaneously. These evaluations are fully independent; the slowdown arises from memory and parallelism limits of the hardware rather than algorithmic constraints. With more cores and threads, we expect this overhead to diminish, so scalability is primarily bounded by system configuration for our protocol. In summary, our protocol matches or outperforms state-of-the-art works that operate on plaintext sets while additionally enabling secure label retrieval and downstream analytics.

C. Experiments on Real-World Datasets

We benchmark our ELSA protocol with downstream logistic regression on three FDB datasets. Since ELSA supports arbi-

TABLE III: ELSA latency results for FDB datasets (per query). Best computation times are in bold. T denotes the number of threads. Sender count and storage refer to the number of senders and storage needed for each sender, respectively. $\delta = 64$ and $\kappa = 8$.

Dataset	Sender		Comm. Time (s)				Online Latency (s)				
	Count	Storage (MB)	20 Gbps	10 Gbps	1 Gbps	500 Mbps	T=1	T=2	T=4	T=8	T=16
VLDP	176	1291.8	9.9	19.8	197.5	395.0	168.1	150.8	92.2	**62.6**	63.2
CCTFD	960	738.2	53.9	107.9	1078.3	2157.4	128.6	104.1	66.9	**63.5**	69.8
IEEECIS	250	1291.9	14.0	28.1	280.7	561.4	163.0	147.9	88.3	**58.9**	62.9

TABLE IV: Runtime percentage of FHE operations for a single query membership and computation in ELSA. $\kappa = 8$ and T=8.

FHE Operation	Party	Runtime %			Req. Depth
		VLDP	CCTFD	IEEECIS	
Query Generation	Receiver	0.75	0.73	0.77	-
VAF w/ Windowing	Sender(s)	56.0	57.93	57.58	23
Label Retrieval	Sender(s)	12.38	6.82	12.78	2
Logistic Reg. / Flag	Leader	30.51	34.17	28.49	13/17
Decryption	Receiver	0.36	0.35	0.38	-

trary computations on encrypted real-valued labels and scales to a very large number of senders, selecting a protocol as a case for comparison is a complex task. We thus microbenchmark ELSA for latency and communication overhead and discuss some of the limitations of prior works for comparison in our setting. Existing FHE-based approaches that enable post-intersection computation, such as [26], [29], [50], operate only over the intersection itself. Other protocols based on SMPC or garbled circuits [31], [37], [51], [61], [71] typically support only limited symmetric functions linear in the set size, lack support for floating-point inputs, or incur prohibitive communication costs for general computations. KTSJ24 does not support label retrieval, and both PEPSI and Cong *et al.* introduce substantial overhead for computing on encrypted floating-point labels due to finite-field FHE, making them unsuitable for the floating-point labels present in the FDB.

Table III shows our protocol's online latency under different numbers of threads, communication cost, and storage needs under varying counts of senders. We also display the communication latency for δ= 64 with κ= 8 under various bandwidths. We vary the number of senders per dataset based on its label distribution and size to ensure balanced partitioning across senders. Total latency for all three datasets remains under 65 sec for δ= 64. Importantly, increasing δ to 128 adds only a modest amount of additional costs for our protocol (with $\kappa = 16$). The online communication remains the same for 128-bit items, and latency is increased by about 15%, 41%, and 11%, respectively, for three datasets. Importantly, increasing the item length to $\delta = 128$ adds only a modest amount of additional. We report the runtime percent of each step of ELSA for 3 datasets in Table IV. The selection stage (VAF evaluation) dominates runtime, accounting for over 50% of end-to-end latency. We note that the number of senders is kept consistent in both Table III and Table IV.

In terms of accuracy, our protocol matches plaintext-level logistic regression up to 24 bits under CKKS. For comparison, we evaluated CKKS performance on a division-heavy task by implementing a privacy-preserving One-Sample T-test ($t = (\bar{x} - \mu)/(s/\sqrt{n})$) on a synthetic dataset ($N = 256$, $\mathcal{N}(0, 1)$). We employed the scale-invariant inverse square root approximation algorithm [100] to compute the standard deviation inverse. The encrypted analytic achieved 16-bit precision compared to plaintext baseline with a 10.53 sec latency. While MPC (via secret sharing or full circuit-based PSI) can achieve exact plaintext accuracy, we do not compare against such baselines as they incur prohibitive communication overhead at the sender scales targeted by ELSA [101]. MPC may be competitive at a very small scale, but it becomes communication-bound as the number of senders grows, whereas our protocol is compute-bound and parallelizes efficiently across senders.

The leader sender additively aggregates the returned label and flag ciphertexts and evaluates the final analytic function in the end-to-end tests. While homomorphic addition itself is inexpensive, profiling with `/usr/bin/time`, we observed that at scale the leader's cost is dominated by CPU and memory pressure from handling large ciphertext aggregates (peak memory reached around 65 GB for CCTFD), rather than storage or network overheads. To reduce the load on a single sender, several methods can be applied. Since addition is associative and commutative, the aggregation can be restructured via tree-based mini-leaders, sharded leaders that each handle disjoint sender/query subsets, or dynamically selected leaders based on current load, thereby reducing single-leader bottlenecks and improving scalability.

IX. Conclusion

In this work, we present an end-to-end protocol for encrypted label selection and analytics (ELSA) on distributed datasets. It combines CKKS-based thresFHE with high-accuracy VAF approximations to support large identifier domains and real-valued labels, enabling encrypted label extraction and efficient downstream computations. We evaluate our method on real-world fraud datasets and demonstrate that our protocol is significantly better than prior work, achieving up to 6.8× speed-up, while maintaining scalability and high precision. In future work, we aim to explore multi-query workloads, richer predicates, and additional system-level optimizations.

Acknowledgments

This research was supported in part by the National Science Foundation under CNS-2337321, OAC-2312973, and the Culture, Sports and Tourism R&D Program through the Korea Creative Content Agency grant funded by the Ministry of Culture, Sports and Tourism in 2024 (RS-2024-00332210). We thank the anonymous reviewers for their valuable comments and Mr. Sungkwang Ji and Dr. Brent ByungHoon Kang from KAIST for helpful discussions.

References

[1] M. T. McCarthy, "Usa patriot act," 2002.
[2] A. Alexandru and K. Rohloff, "Privacy-preserving data sharing across financial institutions," NIST Workshop on Privacy-Enhancing Cryptography, 2024. [Online]. Available: https://csrc.nist.gov/csrc/media/presentations/2024/wpec2024-2a6/images-media/wpec2024-2a6-slides-andreea-kurt--PPDS-financial.pdf
[3] "Anti-money laundering (aml) solution for banks," https://dualitytech.com/use-cases/financial-services-industry/anti-money-laundering-aml, Duality Technologies, accessed: 2025-05-23.

[4] Association of Certified Fraud Examiners, "Report to the nations: 2022 global study on occupational fraud and abuse," 2022.
[5] Federal Financial Institutions Examination Council, "Bank secrecy act/anti-money laundering examination manual," 2021.
[6] "Bank secrecy act of 1970," Public Law 91-508, 84 Stat. 1114, 1970, codified at 31 U.S.C. §§5311–5332.
[7] "Regulation (eu) of the european parliament and of the council (general data protection regulation)," Official Journal of the European Union L119/1, 2016, https://eur-lex.europa.eu/eli/reg/2016/679/oj.
[8] Office of the National Coordinator for Health IT, "21st century cures act: Interoperability, information blocking, and the onc health it certification program," Federal Register 85 FR 25642, 2020.
[9] J. R. Vest and L. D. Gamm, "Health information exchange: Persistent challenges and new strategies," *Journal of the AMIA*, 2010.
[10] National Association of REALTORS®, "Data privacy & security," https://www.nar.realtor/data-privacy-security, 2017, accessed 24 Jul 2025.
[11] N. Wang, W. Zhou, J. Wang, Y. Guo, J. Fu, and J. Liu, "Secure and efficient similarity retrieval in cloud computing based on homomorphic encryption," *IEEE TIFS*, vol. 19, pp. 2454–2469, 2024.
[12] C. Fu and X. Zhang et al., "Label inference attacks against vertical federated learning," in *USENIX Security*, 2022.
[13] J. Liagouris, V. Kalavri, M. Faisal, and M. Varia, "Secrecy: Secure collaborative analytics in untrusted clouds," in *USENIX NDSI*, 2023.
[14] M. Dong, Y. Chen, C. Zhang, Y. Bai, and Y. Cao, "Multi-party private set operations from predicative zero-sharing," *IACR ePrint*, 2025.
[15] Statista, "Number of fdic-insured banks in the u.s. in 2023: 4,470," 2023. [Online]. Available: https://www.statista.com/statistics/184536/number-of-fdic-insured-us-commercial-bank-institutions/
[16] E. Szerdocz. How banks can unite to improve aml detection. [Online]. Available: https://www.partisia.com/docs/collaborative-aml-how-banks-can-unite-to-improve-anti-money-laundering-detection/
[17] M. B. van Egmond, V. Dunning, S. van den Berg, T. Rooijakkers, A. Sangers, T. Poppe, and J. Veldsink, "Privacy-preserving anti-money laundering using secure multi-party computation," in *ICFCDS*, 2024.
[18] C. News, "Banks are willing to cooperate for money laundering investigations," 2025, accessed: 2025-01-17. [Online]. Available: https://www.cbsnews.com/news/tax-evasion-billions-offshore-fatca-tax-reporting-loophole-senate-finance-committee-robert-brockman/
[19] M. Alkhalili, M. H. Qutqut, and F. Almasalha, "Investigation of applying machine learning for watch-list filtering in anti-money laundering," *IEEE Access*, vol. 9, pp. 18 481–18 496, 2021.
[20] Oracle and D. Technologies, "Privacy-protected aml queries: Collaborative aml investigations across banks with encrypted queries," 2025, accessed: 2025-03-07. [Online]. Available: https://www.oracle.com/oce/dc/assets/CONT4168C8AB242C413693AD0413917DCE7E/native/oracle-duality-data-sheet.pdf
[21] J. Takeshita et al., "Provably secure contact tracing with conditional private set intersection," in *SecureComm*. Springer, 2021, pp. 352–373.
[22] P. K. Doppalapudi, P. Kumar, A. Murphy, C. Rougeaux, R. Stearns, S. Werner, and S. Zhang, "The fight against money laundering: Machine learning is a game changer," McKinsey & Company '22, https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/the-fight-against-money-laundering-machine-learning-is-a-game-changer.
[23] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A {Fault-Tolerant} abstraction for {In-Memory} cluster computing," in *USENIX NSDI*, 2012, pp. 15–28.
[24] M. Armbrust, A. Ghodsi, R. Xin, M. Zaharia *et al.*, "Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics," in *Proceedings of CIDR*, vol. 8, 2021, p. 28.
[25] Federal Financial Institutions Examination Council (FFIEC), "Bsa/aml risk assessment: Overview," https://bsaaml.ffiec.gov/manual/BSAAMLRiskAssessment/01, 2021, accessed: 2025-07-13.
[26] K. Cong, R. C. Moreno, M. B. da Gama, W. Dai, I. Iliashenko, K. Laine, and M. Rosenberg, "Labeled psi from homomorphic encryption with reduced computation and communication," in *Proceedings of the 2021 ACM CCS*, 2021, pp. 1135–1150.
[27] R. A. Mahdavi, N. Lukas, F. Ebrahimianghazani, T. Humphries, B. Kacsmar, J. Premkumar, X. Li, S. Oya, E. Amjadian, and F. Kerschbaum, "Pepsi: Practically efficient private set intersection in the unbalanced setting," in *USENIX Security*, 2024.
[28] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, "Efficient circuit-based psi via cuckoo hashing," in *EUROCRYPT*, 2018.
[29] Y. Son and J. Jeong, "Psi with computation or circuit-psi for unbalanced sets from homomorphic encryption," in *ACM ASIACCS*, 2023.
[30] N. Chandran, D. Gupta, and A. Shah, "Circuit-psi with linear complexity via relaxed batch opprf," *PoPETS*, 2022.
[31] J. Gao, N. Trieu, and A. Yanai, "Multiparty private set intersection cardinality and its applications," *PoPETS*, 2024.
[32] P. Rindal and P. Schoppmann, "Vole-psi: fast oprf and circuit-psi from vector-ole," in *EUROCRYPT*. Springer, 2021.
[33] P. Buddhavarapu, A. Knox, P. Mohassel, S. Sengupta, E. Taubeneck, and V. Vlaskin, "Private matching for compute," *IACR ePrint*, 2020.
[34] T. Lepoint et al., "Private join and compute from pir with default," in *ASIACRYPT*. Springer, 2021, pp. 605–634.
[35] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, and M. Yung, "On deploying secure computing: Private intersection-sum-with-cardinality," in *2020 EUROS&P*.
[36] N. Koirala, J. Takeshita, J. Stevens, and T. Jung, "Summation-based private segmented membership test from threshold-fully homomorphic encryption," *PoPETS*, 2024.
[37] D. Mouris, D. Masny, N. Trieu, S. Sengupta, P. Buddhavarapu, and B. Case, "Delegated private matching for compute," *PoPETS*, 2024.
[38] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *ASIACRYPT*. Springer, 2017, pp. 409–437.
[39] J. H. Cheon, W. Kim, and J. H. Park, "Efficient homomorphic evaluation on large intervals," *IEEE TIFS*, vol. 17, pp. 2553–2568, 2022.
[40] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, and A. Shah, "Efficient linear multiparty psi and extensions to circuit/quorum psi," in *ACM CCS*, 2021.
[41] O. Nevo, N. Trieu, and A. Yanai, "Simple, fast malicious multiparty private set intersection," in *ACM CCS*, 2021.
[42] M. Wu, T. H. Yuen, and K. Y. Chan, "{O-Ring} and {K-Star}: Efficient multi-party private set intersection," in *USENIX Security*, 2024.
[43] J. Vos, S. Pentyala, S. Golob, R. Maia, D. Kelley, Z. Erkin, M. De Cock, and A. Nascimento, "Privacy-preserving membership queries for federated anomaly detection," *PoPETS*, 2024.
[44] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *NDSS*, 2012.
[45] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 351–371.
[46] P. Hallgren, C. Orlandi, and A. Sabelfeld, "Privatepool: Privacy-preserving ridesharing," in *IEEE CSF*. IEEE, 2017, pp. 276–291.
[47] B. Kreuter, "Secure multiparty computation at google. real world crypto," 2017.
[48] M. Yung, "From mental poker to core business: Why and how to deploy secure computation protocols?" in *ACM CCS*, 2015, pp. 1–2.
[49] H. Son, S. Paik, Y. Kim, S. Kim, H. Chung, and J. H. Seo, "Doubly efficient fuzzy private set intersection for high-dimensional data with cosine similarity," *IACR ePrint*, 2025.
[50] H. Chen, Z. Huang, K. Laine, and P. Rindal, "Labeled psi from fully homomorphic encryption with malicious security," in *Proceedings of the 2018 ACM CCS*, 2018, pp. 1223–1237.
[51] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai, "Efficient circuit-based psi with linear communication," in *EUROCRYPT*, 2019.
[52] Y. Yang, X. Dong, Z. Cao, J. Shen, R. Li, Y. Yang, and S. Dou, "Empsi: Efficient multiparty private set intersection (with cardinality)," *Frontiers of Computer Science*, vol. 18, no. 1, p. 181804, 2024.
[53] L. Kissner and D. Song, "Privacy-preserving set operations," in *CRYPTO*. Springer, 2005, pp. 241–257.
[54] X. Yang, L. Cai, Y. Wang, K. Yin, L. Sun, and J. Hu, "Efficient unbalanced quorum psi from homomorphic encryption," in *ACM ASIACCS*, 2024, pp. 1003–1016.
[55] J. Gao, S. Nguyen, and N. Trieu, "Toward a practical multi-party private set union," *IACR ePrint*, 2023.
[56] C. Zhang, Y. Chen, W. Liu, M. Zhang, and D. Lin, "Linear private set union from multi-query reverse private membership test," in *USENIX Security*, 2023, pp. 337–354.
[57] J. Vos, M. Conti, and Z. Erkin, "Fast multi-party private set operations in the star topology from secure ands and ors," *IACR ePrint*, 2022.
[58] X. Liu and Y. Gao, "Scalable multi-party private set union from multi-query secret-shared private membership test," in *ASIACRYPT*. Springer, 2023, pp. 237–271.

[59] Y. Jia, S.-F. Sun, H.-S. Zhou, and D. Gu, "Scalable private set union, with stronger security," in *USENIX Security*, 2024, pp. 6471–6488.

[60] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *ACM ASIACCS*, 2012, pp. 85–86.

[61] T. Duong, D. H. Phan, and N. Trieu, "Catalic: Delegated psi cardinality with applications to contact tracing," in *ASIACRYPT*, 2020.

[62] A. Abadi, S. Terzis, and C. Dong, "Feather: Lightweight multi-party updatable delegated private set intersection," *IACR ePrint*, 2020.

[63] Y. Yang, Y. Yang, X. Chen, X. Dong, Z. Cao, and J. Shen, "Dmpsi: Efficient scalable delegated multiparty psi and psi-ca with oblivious prf," *IEEE Transactions on Services Computing*, 2024.

[64] S. Garg, M. Hajiabadi, A. Jain, Z. Jin, O. Pandey, and S. Shiehian, "Credibility in private set membership," in *IACR PKC*, 2023.

[65] E. Chielle, H. Gamil, and M. Maniatakos, "Real-time private membership test using homomorphic encryption," in *DATE*. IEEE, 2021.

[66] S. Ramezanian, T. Meskanen, M. Naderpour, V. Junnila, and V. Niemi, "Private membership test protocol with low communication complexity," *Digital Communications and Networks*, 2020.

[67] A. Kulshrestha and J. Mayer, "Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation," in *USENIX Security*, 2021, pp. 893–910.

[68] K. C. Wang and M. K. Reiter, "How to end password reuse on the web," *arXiv preprint arXiv:1805.00566*, 2018.

[69] Y. Chen, M. Zhang, C. Zhang, M. Dong, and W. Liu, "Private set operations from multi-query reverse private membership test," in *IACR PKC*. Springer, 2024, pp. 387–416.

[70] K. Chida, K. Hamada, A. Ichikawa, M. Kii, and J. Tomida, "Communication-efficient inner product private join and compute with cardinality," in *ACM ASIACCS*, 2023, pp. 678–688.

[71] K. Han, S. Kim, and Y. Son, "Private computation on common fuzzy records," *PoPETS*, 2025.

[72] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, 2009.

[73] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *Journal of the ACM (JACM)*, 2013.

[74] C. Gentry, *A fully homomorphic encryption scheme*. Stanford university, 2009.

[75] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM TOCT*, 2014.

[76] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *CRYPTO*. Springer, 2012, pp. 868–886.

[77] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR ePrint*, 2012.

[78] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *JoC*, 2020.

[79] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. Rasmussen, and A. Sahai, "Threshold cryptosystems from threshold fully homomorphic encryption," in *CRYPTO*. Springer, 2018, pp. 565–596.

[80] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold fhe," in *EUROCRYPT*, 2012.

[81] E. Kim, J. Jeong, H. Yoon, Y. Kim, J. Cho, and J. H. Cheon, "How to securely collaborate on data: Decentralized threshold he and secure key update," *IEEE Access*, vol. 8, pp. 191 319–191 329, 2020.

[82] S. Badrinarayanan, P. Miao, S. Raghuraman, and P. Rindal, "Multi-party threshold private set intersection with sublinear communication," in *IACR PKC*. Springer, 2021, pp. 349–379.

[83] G. Jiang, H. Zhang, J. Lin, F. Kong, and L. Yu, "Optimized verifiable delegated private set intersection on outsourced private datasets," *Computers & Security*, vol. 141, p. 103822, 2024.

[84] K. EdalatNejad, M. Raynal, W. Lueks, and C. Troncoso, "Private collection matching protocols," *arXiv preprint arXiv:2206.07009*, 2022.

[85] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *IEEE S&P*. IEEE, 2013, pp. 463–477.

[86] D. Natarajan, A. Loveless, W. Dai, and R. Dreslinski, "Chex-mix: Combining homomorphic encryption with trusted execution environments for oblivious inference in the cloud," in *IEEE EuroS&P*, 2023.

[87] J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," in *ASIACRYPT*. Springer, 2020.

[88] S. Martin, N. Koirala, H. Berens, T. Rozgonyi, M. Brody, and T. Jung, "Hydia: Fhe-based facial matching with hybrid approximations and diagonalization," *PoPETS*, 2025.

[89] S. Chatel, C. Knabenhans, A. Pyrgelis, C. Troncoso, and J.-P. Hubaux, "Veritas: Plaintext encoders for practical verifiable homomorphic encryption," in *ACM CCS*, 2024, pp. 2520–2534.

[90] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *ACM CCS*, 2017, pp. 1243–1255.

[91] N. P. Smart and F. Vercauteren, "Fully homomorphic simd operations," *Designs, codes and cryptography*, vol. 71, pp. 57–81, 2014.

[92] A. R. T. Donders, G. J. Van Der Heijden, T. Stijnen, and K. G. Moons, "A gentle introduction to imputation of missing values," *Journal of clinical epidemiology*, vol. 59, no. 10, pp. 1087–1091, 2006.

[93] D. Vatsalan, Z. Sehili, P. Christen, and E. Rahm, "Privacy-preserving record linkage for big data: Current approaches and research challenges," *Handbook of big data technologies*, pp. 851–895, 2017.

[94] B. Pan, N. Stakhanova, and S. Ray, "Data provenance in security and privacy," *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1–35, 2023.

[95] Y. Zhang, A. O'Neill, M. Sherr, and W. Zhou, "Privacy-preserving network provenance," *VLDB*, vol. 10, no. 11, pp. 1550–1561, 2017.

[96] M.-I. Pleşa and R. F. Olimid, "Privacy-preserving multi-party search via homomorphic encryption with constant multiplicative depth," *IACR ePrint*, 2024.

[97] J. Lauinger, J. Ernstberger, A. Finkenzeller, and S. Steinhorst, "Janus: Fast privacy-preserving data provenance for tls," *PoPETS*, 2025.

[98] A. Al Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee *et al.*, "Openfhe: Open-source fully homomorphic encryption library," in *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2022, pp. 53–63.

[99] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, "Homomorphic encryption security standard," HomomorphicEncryption.org, Toronto, Canada, Tech. Rep., November 2018.

[100] J. Moon, Z. Omarov, D. Yoo, Y. An, and H. Chung, "Adaptive successive over-relaxation method for a faster iterative approximation of homomorphic operations," *Cryptology ePrint Archive*, 2024.

[101] R. De Viti, I. Sheff, N. Glaeser, B. Dinis, R. Rodrigues, B. Bhattacharjee, A. Hithnawi, D. Garg, and P. Druschel, "{CoVault}: Secure, scalable analytics of personal data," in *USENIX Security*, 2025.

[102] N. Koirala, S. Paik, S. Martin, H. Berens, T. Januszewicz, J. Takeshita, J. H. Seo, and T. Jung, "Select-then-compute: Encrypted label selection and analytics over distributed datasets using fhe (extended version)," *IACR ePrint*, 2025.

Appendix A

A. Parameter Selection for VAFs

We provide a parameter selection method for our new VAF via grid search, along with presets used for experiments.

The VAF is instantiated by tuning:

- wDEP parameters: expansion rate L, initial domain range R, and number of domain extensions n_{DEP}.
- Bell-shaped function parameters: number of transformations n_T and number of squarings n_{SQ}.

Suppose that our goal is to design a VAF for the domain $[-M, M]$ with an error of at most ν. If we denote ϵ_{sep} and $(B, \epsilon_{\text{BS}})$ as the separation factor of the resulting wDEP and the parameter for the resulting bell-shaped function, respectively, then these parameters should satisfy the following conditions:

1) $L^{n_{\mathsf{DEP}}} R \geq M$ to ensure $[-M, M] \subset [-L^{n_{\mathsf{DEP}}} R, L^{n_{\mathsf{DEP}}} R]$
2) $B \leq \nu$ to ensure sufficient accuracy.
3) $\epsilon_{\text{BS}} \leq \epsilon_{\text{sep}}$ to ensure the condition of Theorem 1.

From these requirements, we can derive several relationships between parameters. The first condition gives $n_{\mathsf{DEP}} > \frac{\log_2 M - \log_2 R}{\log_2 L}$, which can be determined once R is chosen. For the second condition, note that the error bound B depends on n_{SQ}, namely, $2^{-2^{n_{\mathsf{SQ}}+1}}$. Hence, $2^{n_{\mathsf{SQ}}+1} \geq -\log_2 \nu$ suffices.

Parameters: Each sender $i \in [1, n-1]$ receives encrypted inputs $(c_{x_i}, c_{\ell b_i})$, where X_i and ℓb_i are the input set and associated labels, and y is the receiver's input. λ: computational security param., δ: bit-length of elements; both are public. Slot-wise windowing parameter $\kappa \in \mathbb{N}$ and two VAFs, $f^{\text{int}}_{\text{VAF}}$ and $f^{\text{agg}}_{\text{VAF}}$, defined over domains $[-2^{\delta/\kappa}, 2^{\delta/\kappa}]$ and $[-n, n]$ and the bound error ν, respectively, are known publicly. Senders posses $f(\{c_{\ell b_i}\}_{i \in L}; \Theta)$, where Θ is some private parameter and L contains the labels associated with y.

1. [**Parameters**]
   a. **Threshold FHE**: Parties, including the receiver and senders, agree on parameters $(N, q, \delta, D, \alpha, \lambda)$ for the thresFHE CKKS scheme.
   b. **Key Distribution and Setup**: $\alpha - 1$ senders and receiver run a SMPC protocol *ThresFHE.KeyGen* that provides each partial key shareholder $i \in [1, \alpha)$ a secret key share $sk \in \{sk_1, sk_2, \ldots, sk_\alpha\}$. Common public key pk and evaluation key evk are broadcast publicly. Senders jointly agree on a permutation, obtaining $perm_i$ containing 1 in ${\rho_i}^{\text{th}}$ slot and 0 elsewhere. All parties, including data owners, have access to pk and evk after this step.
2. [**Encryption**]
   a. Each data owner j such that $j \in \{1, \ldots, n-1\}$, organizes its input vectors $x \in X_j$ and corresponding labels $\ell \in \ell b_j$ into $\mathbb{R}^m$. These vectors are then divided into κ parts, which are then sequentially packed into κ plaintexts of size $N/2$, producing $\kappa \times \frac{2m}{N}$ plaintexts, encrypted using *ThresFHE.Enc*. They also compute the statistics ℓb_{stat}. Every data owner j encrypts its sets, and the statistics value independently and forwards the resulting ciphertexts to the appropriate senders $i \in \{1, \ldots, n-1\}$. After receiving the ciphertext pair(s) $(c_{x_i}, c_{\ell b_i})$ and the statistics ciphertext $c_{\ell b_{\text{stat}}}$, sender i may hold multiple ciphertexts if $|X_i| > \eta$.
   b. **Encrypt replicas of** y: Receiver constructs a vector of length η, with each element $y \in \mathbb{R}$. Then, it divides the vector into κ segments, sequentially packs the first η elements into a plaintext, and encrypts it using *ThresFHE.Enc*. As a result, each consecutive κ slots of the resulting ciphertext c_y contains a single y value.
3. [**Compute Intersection**]
   For each pair $(c_{x_i}, c_{\ell b_i})$, and the ciphertext c_y, sender i uses *ThresFHE.Eval* with evk to execute the following steps:
   a. Computes $\textit{diff}_i = c_y - c_{x_i}$. Next, it applies a piecewise function $\textit{select}_i = f^{\text{int}}_{\text{VAF}}(\textit{diff}_i)$ using evk and obtains $\textit{select}_i$. If possessing multiple set ciphertexts, the sender computes multiple $\textit{select}_i$(s) in parallel.
   b. Applies rotation-and-multiplication technique to to multiply consecutive κ slots in each $\textit{select}_i$.
   c. Homomorphically multiplies a masking vector that comprised by $\frac{\eta}{2\kappa}$ concatenations of $(1, 0, \ldots, 0) \in \mathbb{R}^\kappa$ in each $\textit{select}_i$.
4. [**Label(s) Retrieval and (Optional) Permutation**]
   For each pair $(c_{x_i}, c_{\ell b_i})$ and $\textit{select}_i$, sender i homomorphically executes the following:
   a. Computes $\textit{resLab}_i = \textit{select}_i \times c_{\ell b_i}$ where $c_{\ell b_i}$ corresponds to c_{x_i}. If possessing multiple $\textit{resLab}_i$, the sender sums them to a single $\textit{resLab}_i$.
   b. Rotates and adds $\textit{resLab}_i$ and $\textit{select}_i$ with $\log(\eta)$ operations to replicate intersection label and indicator for intersection in all slots. The result from the latter is set to $\textit{flag}_i$. Then it computes $\textit{resLab}_i + (\vec{1} - \textit{flag}_i) \times c_{\ell b_{\text{stat}}}$ where $\vec{1} \in \mathbb{R}^{N/2}$ is a vector filled with 1.
   c. If f is a symmetric function sender computes $\textit{resLab}_i \times \textit{perm}_i$, where $\textit{perm}_i$ permutes, and isolates label in a single slot of $\textit{resLab}_i$. Otherwise, sender ignores this step.

   Finally, each sender transmits $\textit{resLab}_i$ and $\textit{flag}_i$ to *leader*.
5. [**Computation on Retrieved Label(s)**]
   The leader sender obtains $\{\textit{resLab}_i, \textit{flag}_i\}_{i \in (n-1)}$ from all the senders and computes the following:
   a. Computes $\textit{resLab} = \sum_{i=0}^{n-1} \textit{resLab}_i$. Then, using evk and private parameter Θ, the leader computes $\mathbf{z} = f(\textit{resLab}_i; \Theta)$, where f operates labels located at the slots of *resLab*.
   b. Computes $\overline{\textit{flag}} = \sum_{i=0}^{n-1} \textit{flag}_i$ and homomorphically evaluates $\textit{flag} = f^{\text{agg}}_{\text{VAF}}(\overline{\textit{flag}})$.

   The leader sender returns $\mathbf{z}$ and *flag* to the receiver and $\alpha - 1$ senders for partial decryption.
6. [**Partial and Final Decryption**]
   a. **Partial decryption & smudging noise** : Upon receiving z and *flag*, each partial key-share holder sender $i \in [1, \alpha)$ uses their secret key share sk_i to partially decrypt them separately using *ThresFHE.PartialDec*, and introduces a smudging noise e_{smg} to the partial decryptions $part_i$ and $part_{flag_i}$. Each sender i computes $part_i$ and $part_{flag}$ and sends their decryption share to the receiver.
   b. **Final decryption**: The receiver uses their partial decryption key sk_i and z and $flag$ to execute *ThresFHE.PartialDec* and obtains their partial decryption share $part_i$ and $part_{flag_i}$. Receiver combines their share with partial decryptions received from the $\alpha - 1$ senders to compute $\{part_i : sk_i\}_{i \in [0, \alpha)}$ and $\{part_{flag_i} : sk_i\}_{i \in [0, \alpha)}$ using *ThresFHE.Combine* and pk to obtains the result vector of length η.
   c. **Interpretation of result**: If the decryption of *flag* is 0, then the receiver rejects the result from the decryption of z. Otherwise, the receiver confirms the result of $f(\{\ell b_i\}_{i \in L}; \Theta)$.

Fig. 7: Full ELSA protocol without a Trusted Third Party

To ensure the third condition, we utilize numerical algorithms to analyze each ϵ_{sep} and ϵ_{BS}. For the former, we leverage the fact that our base function $D(x; k) = \frac{3\sqrt{3}}{2k\sqrt{k}}(k - x^2)$ is monotonic for the domain $[-\sqrt{k/3}, \sqrt{k/3}]$. In addition, according to the domain extension process, $\epsilon_T = D(L)$ holds. Thus, we conclude $\epsilon_{\text{sep}} = \min\{D(L), f_{\text{wDEP}}(1)\}$. For the latter, we estimate ϵ_{BS} by solving the root-finding problem $f_{\text{BS}}(x) = B$ nearby 0. We found that a simple bisection search method is enough for our purpose.

To sum up, once the domain $[-M, M]$, the precision ν, and the base function for the wDEP are fixed, we can conduct a grid search on parameters by the following pipeline.

TABLE V: Parameter Presets for Each Domain Size

Parameter	2^1	2^2	2^4	2^5	2^6	2^8	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}
k	N/A	N/A	N/A	17	17	17	6.75	6.75	6.75	17	17	6.75
L	N/A	N/A	N/A	4	4	4	2.59	2.59	2.59	4	4	2.59
R	2	4	16	4	11	4	158.54	91.09	148.45	5112.73	73139	12583
n_{DEP}	0	0	0	2	1	3	2	4	5	2	1	5
n_{VAF}	4	7	4	3	4	4	8	7	8	16	20	16
newVAF	F	F	T	T	T	T	T	T	T	T	T	T
depth	7	10	13	16	15	19	22	25	28	32	35	38

1) Make a grid for possible $R \in [1, M]$. Then we obtain the corresponding n_{DEP} to satisfy the condition (1).
2) Compute ϵ_{sep} from the given weak DEP parameters.

Algorithm 2 Depth-Efficient Transformation DETBS

```
Require: Input $x \in \mathbb{R}$ and the number of transformations $n \in \mathbb{N}$,
    and transformation parameters $\frac{a}{q}, \frac{b}{q} \in \mathbb{Q}$
Ensure: Result $y$ from applying $x \mapsto \frac{a}{q}x + \frac{b}{q}$ $n$ times.
1: Initialize $Q \leftarrow 1$.
2: for idx from 1 to $n-1$ do do
3:     Compute $x \leftarrow (ax + Qb)^2$.
4:     Update $Q \leftarrow Q^2 \cdot q^2$.
5: end for
6: Compute $x \leftarrow \frac{1}{Q \cdot q}(ax + Qb)$ and $y \leftarrow x^2$.
7: return $y$.
```

3) Choose n_{SQ} to satisfy the condition (2) and find the suitable n_T to satisfy the condition (3) through solving the root-finding algorithm $f_{\mathrm{BS}}(x) = B$.

Once parameters are found, we select the parameter requiring the smallest multiplicative depth. If there are ties, we select one with a smaller n_T because the transformation for the bell-shaped function is cheaper than the domain extension. We provide the parameter presets in Table V. The implementation of the grid search algorithm can be found in our source code.

B. Algorithms

Fused DEP Iteration: We detail the algorithms for the improved VAFs given in Section VI-A. We first present the depth-efficient transformation method DETBS in Algorithm 2.

Theoretically, we can use algorithm DETBS to design the whole VAF. However, we can observe that the intermediate value in x exponentially grows with respect to the number of iterations n, which may result in overflow during evaluation. For this reason, we restrict the maximum number of fused transformations per one DETBS, say $n_{\max}$. When evaluating the VAF with transformations $n > n_{\max}$, we first divide n by $n_{\max}$ to get the quotient and remainder, say (q, r). Then we run DETBS with $n_{\max}$ q times and after evaluation, we run DETBS with r on the output. We describe the above process in Algorithm 3, called VAFfromBS.

Finally, we merge the VAFfromBS algorithm to the domain extension process, with some tricks to reduce the depth. First, we slightly tweak the order of computation when evaluating the base function $D(x; k) = \frac{3\sqrt{3}}{2k\sqrt{k}}x(k - x^2)$, namely, $y_1 \leftarrow k - x^2$; $y_2 \leftarrow \frac{3\sqrt{3}}{2k\sqrt{k}}x$; $y \leftarrow y_1 \cdot y_2$. In particular, we can save more levels at the domain extension process, i.e., $y \mapsto L^i D(\frac{y}{L^i})$ because we can think of $\frac{y}{L^i}$ as $\frac{1}{L^i} \cdot L^{i+1} D(\frac{\hat{y}}{L^{i+1}})$, where $\hat{y}$ is the input of previous domain extension processes. That is, we can tweak the domain extension process as $y \leftarrow LD(y)$ with some careful scaling at the beginning. In addition, multiplication by L at the end can be applied by letting $y_2 \leftarrow \frac{3\sqrt{3}L}{2k\sqrt{k}}x$. Hence, we can compute the domain extension process using 2 levels per iteration.

In addition, we also observed that when we combine the weak DEP and our VAF, we can save one level when calculating $(1 - \frac{3}{2}x^2)^2$ by multiplying $\sqrt{3/2}$ to y_2 at the last domain extension process. We describe the complete VAF construction in Algorithm 4.

Algorithm 3 VAF from Bell-Shaped Function (VAFfromBS)

```
Require: Input $x \in \mathbb{R}$, the number of transformations $n \in \mathbb{N}$,
    transformation parameters $\frac{a}{q}, \frac{b}{q} \in \mathbb{Q}$, and the maximum number
    of fused transformations $n_{\max}$.
Ensure: Result $y$ from applying $x \mapsto \frac{a}{q}x + \frac{b}{q}$ $n$ times.
1: Compute $q, r \in \mathbb{N}$ such that $n = q \cdot n_{\max} + r$ and $0 \le r < n_{\max}$.

2: for idx from 1 to $q$ do do
3:     Compute $x \leftarrow \mathsf{DETBS}(x, n_{\max}, \frac{a}{q}, \frac{b}{q})$
4: end for
5: Compute $y \leftarrow \mathsf{DETBS}(x, r, \frac{a}{q}, \frac{b}{q})$.
6: return $y$.
```

Algorithm 4 VAF from Weak DEP and Bell-Shaped Functions

```
Require: Input $x \in \mathbb{R}$, wDEP parameter $k$, number of extensions
    $n_{\mathrm{DEP}} \in \mathbb{N}$, extension rate $L$, base domain range $R$, and
    $n_T, n_{\mathrm{SQ}} \in \mathbb{N}$ for the number of transformations and squarings
    for bell-shaped function.
Ensure: The VAF evaluation result $f^{(n)}(x; k)$.
1: Set $y \leftarrow \frac{x}{L^{n-1}R}$.
2: for $idx$ from 1 to $n_{\mathrm{DEP}}$ do
3:     Compute $\tilde{y} \leftarrow (k - y^2)$
4:     If $idx < n_{\mathrm{DEP}}$, then set $y \leftarrow \frac{3\sqrt{3}}{2k\sqrt{k}}Ly$; else, $y \leftarrow \frac{9}{2\sqrt{2}k\sqrt{k}}y$.
5:     Compute $y \leftarrow y \cdot \tilde{y}$.
6: end for
7: Set $y \leftarrow (1 - y^2)^2$
8: $y \leftarrow \mathsf{VAFfromBS}(y, n_T, \frac{3}{2}, -\frac{1}{2})$ :
9: for $idx$ from 1 to $n_{\mathrm{SQ}}$ do do
10:     Compute $y \leftarrow y^2$
11: end for
12: return $y$
```

APPENDIX B

We present the security proofs of the proposed protocol. Due to space constraints, we only provide a sketch; full proofs are available in our extended version [102].

A. Security Proof of the Proposed Protocol

We prove the security of our protocol, denoted π_{ELSA}, in the semi-honest model. Our proof relies on the semantic and simulation security of thresFHE (see [79], [82]). Informally, these properties guarantee ciphertext indistinguishability under the decryption threshold and the existence of a simulator for partial decryptions, respectively. For more details, please refer to Definition 5.4 & Definition 5.5 in [79] or Section 2.5 in [82]. With these definitions, we prove the following statement.

Theorem 2. *Assuming the underlying thresFHE satisfies semantic security and simulation security, π_{ELSA} is a secure implementation of the ideal functionality of Π_{ELSA} as described in Figure 2 according to Definition 1.*

Proof Sketch. We must show the existence of a simulator for the four different types of adversaries possible:

1) A number of the senders $\{\mathcal{S}_n\}$ are corrupted, excluding the leader and receiver.
2) A number of senders are corrupted $\{\mathcal{S}_{n-1}\}$, including the leader $\mathcal{S}_n$, but the receiver is not corrupted.

3) The receiver $\mathcal{R}$ is corrupted along with a subset of the senders $\{\mathcal{S}_n\}$, which does not include the leader sender.
4) The receiver $\mathcal{R}$ is corrupted along with a subset of senders $\{\mathcal{S}_{n-1}\}$ which does include the leader sender $\mathcal{S}_n$.

As we assumed the setup is secure (via SMPC or trusted setup) and the data owner remains offline after outsourcing, we restrict our simulation to the querying and retrieval phases. Due to space limits, we provide simulator sketches here; full constructions are available in the extended version [102].

Case 1 & Case 2: In these cases, since the receiver $\mathcal{R}$ is not corrupted, corrupted parties cannot see the decrypted plaintext of the FHE evaluation circuit. That is, it is enough for the simulator $\mathcal{P}$ to generate thresFHE ciphertexts that appeared during the protocol, e.g., encrypted query c_y, label extraction results $\{resLab_i, flag_{sim}\}$, or the function evaluation result $(z, \overline{flag})$. Here, thanks to the semantic security of the thresFHE, all these ciphertexts are indistinguishable from random, i.e., $\mathcal{P}$ can simulate them via random strings.

Case 3 & Case 4: In these cases, since $\mathcal{R}$ is now corrupted, the simulator $\mathcal{P}$ should simulate the partial decryptions from the function evaluation result f_{eval} and the flag bit $b \in \{0,1\}$. To this end, $\mathcal{P}$ conducts the following procedure:

1) $\mathcal{P}$ computes $flag_{sim} \leftarrow$ thresFHE.$Enc(pk, \vec{b})$, where $\vec{b}$ is a plaintext vector whose all slots are filled by b.
2) $\mathcal{P}$ computes $z_{sim} \leftarrow$ thresFHE.$Encrypt(pk, f_{eval})$.
3) $\mathcal{P}$ first randomly selects $(\alpha - 1)$ senders. Among them, the corrupted senders will generate partial decryptions $\{(p_i, q_i)\}$. On the other hand, for non-corrupted senders, $\mathcal{P}$ generates random ciphertexts $\{(p_{sim}, q_{sim})\}$ and chooses the final ciphertext pair (p_1, q_1) so that $f_{eval} =$ thresFHE.Combine$(pk, \{p_i\} \cup \{p_{sim}\} \cup p_1 \cup p_{\mathcal{R}})$ and $\vec{b} =$ thresFHE.Combine$(pk, \{q_i\} \cup \{q_{sim}\} \cup q_1 \cup q_{\mathcal{R}})$, where $(p_{\mathcal{R}}, q_{\mathcal{R}})$ is the receiver's partial decryption.

Here, the third step is well-defined because at least one selected sender by $\mathcal{P}$ is non-corrupted. Due to the simulation security of the thresFHE, the simulated partial decryptions are indistinguishable from those of real protocols. The remaining communications are the thresFHE ciphertexts that will not be decrypted during the protocol; thus, as was done in Case 1 & Case 2, they can be simulated via random strings.

As in all cases, $\mathcal{P}$ can simulate the behavior of protocol π_{ELSA} in the presence of at most $(\alpha - 1)$ colluding parties, completing the proof. □

B. Proving Provenance Security in the Proposed Protocol

To define the provenance security, we consider a game where the adversary (colluding parties) attempts to distinguish two protocols via the *view* of colluding parties: the *normal* execution of the protocol and the *permuted* protocol when the inputs are randomly permuted across senders. If the adversary cannot distinguish these two cases, then we can ensure that colluding parties cannot learn *which* party contributed *which* data. We provide the formal definition in Definition 4.

Definition 4 (Provenance Security). *Let Π be a n-ary protocol for a receiver $\mathcal{R}$ and senders $\mathcal{S}_1, \ldots, \mathcal{S}_{n-1}$ and $\Pi_{\text{Perm},\delta}$ be a protocol defined by permuting the inputs of senders through the permutation δ over indices $[n-1]$. Then we say that Π satisfies provenance security under the semi-honest model with at most $(\alpha-1)$ collusion if the following inequality holds: For all PPT adversary $\mathcal{A}$ and colluding parties $\mathcal{P}^* \subset \{\mathcal{R}, \mathcal{S}_1, \ldots, \mathcal{S}_{n-1}\}$ such that $|\mathcal{P}^*| \leq \alpha - 1$, and a security parameter λ,*

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}(tr) = 1 \,\middle|\, tr \leftarrow \mathsf{View}_{\Pi}(\mathcal{P}^*)\ \right] - \\ \Pr\left[\mathcal{A}(tr) = 1 \,\middle|\, \begin{array}{c} \sigma \xleftarrow{\$} \text{Perm}_{n-1}, \\ tr \leftarrow \mathsf{View}_{\Pi_{\text{Perm},\sigma}}(\mathcal{P}^*) \end{array} \right] \end{array} \right| < \mathsf{negl}(\lambda),$$

where Perm_{n-1} is the set of all permutations over $[n-1]$, $\mathsf{View}_{\Pi}(\mathcal{S})$ is the union of the transcripts received by parties $\mathcal{P} \in \mathcal{S}$ during the execution of Π.

We prove ELSA satisfies provenance security under at most $(\alpha - 1)$ collusion when evaluating the symmetric functions or the identity function. For the latter, we require that all the labels have the same type; otherwise, the receiver can guess the provenance through the type of the retrieved label.

Theorem 3. *Let the function f be either symmetric or the identity function. Then π_{ELSA} for evaluating f satisfies provenance security with at most $(\alpha - 1)$ collusions.*

Proof Sketch. Given the definition of provenance security, considering the following two cases suffices:

1) The receiver $\mathcal{R}$ is not included in $\mathcal{P}^*$
2) The receiver $\mathcal{R}$ is included in $\mathcal{P}^*$

Case 1: In this case, since $\mathcal{R}$ is not corrupted, the view of the adversary $\mathcal{A}$ includes ciphertexts only, and $\mathcal{A}$ cannot access its decryptions. Thus, due to the semantic security of the thresFHE, $\mathcal{A}$'s view is indistinguishable from random strings of the same length, and any permutation of inputs does not change the distribution. This implies the provenance security.

Case 2: In this case, now $\mathcal{A}$ views the protocol outputs $(\overline{flag}, z)$, along with the ciphertexts communicated during the protocol. For ciphertexts that will not be decrypted, by using the same argument as Case 1, we can ensure that the permutation of the inputs does not affect the distribution. On the other hand, for the protocol outputs, we first note that $\overline{flag}$ contains a bit whether the queried item belongs to the sender's ID set, which is invariant with the input permutation.

For the function evaluation result $z = f_{eval}$, we can check that input permutation does not leak provenance in our choices of f as follows. If f is symmetric, then nothing to prove, since the input permutation does not affect the output. In contrast, if f is an identity function, which is asymmetric, we can utilize the following observation: if a permutation $\sigma \in \text{Perm}_{n-1}$ is applied, then S_i contributes $\rho_{\sigma_i}^{\text{th}}$ slot of $resLab$ instead of ρ_i^{th} slot. That is, we can think of applying σ as a group action over Perm_{n-1} defined by $\rho \mapsto \rho \circ \sigma$. Since this mapping is one-to-one, it preserves uniform distribution over Perm_{n-1}. Since labels share the same type, permuting them leaks no provenance. Thus, the views of corrupted parties remain indistinguishable (with or without permutation).

In all cases, π_{ELSA} satisfies provenance security, and therefore, the protocol must satisfy provenance security. □

Appendix C
Artifact Appendix

A. Description & Requirements

a) How to access.: The artifact for reproducing the results in the paper is at https://doi.org/10.5281/zenodo.17849201. The source code of our implementation and experiments is available at https://github.com/nd-dsp-lab/elsa_protocol.

b) Hardware Requirements.: Our implementation targets a 64-bit CPU-only environment and does not require a GPU. We recommend at least 16 GB RAM for larger domain sizes (e.g., 2^{16} and above), and 5 GB of free disk space for OpenFHE build artifacts, binaries, datasets, and intermediate outputs. Our experiments were run on a single Intel Xeon(R) Gold 5412U server with 512 GB RAM. Note that homomorphic encryption runtimes and memory usage grow with parameter choices and domain sizes, so additional memory headroom may be needed for very large domains (e.g., 2^{128}).

c) Software Requirements.: The implementation requires a Unix-like environment (Linux or macOS), with a C++17-capable toolchain (`gcc` or `clang`), `cmake`, and `make`. The system must include OpenFHE v1.2.3 (from `openfhe-development`.[3]), along with standard utilities like `git` and common POSIX tools. We recommend a recent Linux distribution (e.g., Ubuntu 22.04) for ease of installation and compatibility. If a containerized workflow is used (recommended), a recent Docker installation is also required.

B. Artifact Installation Instructions

We provide the native (non-containerized) instructions for installing OpenFHE, compiling the ELSA protocol, and executing both the individual modules and the complete end-to-end pipeline. While native execution is supported, we strongly recommend the Docker-based workflow for portability and reproducibility. For containerized setup and usage details, refer to the Docker installation instructions.

Install OpenFHE v1.2.3

1) Clone the OpenFHE repository:
```
git clone https://github.com/openfheorg/
openfhe-development.git
cd openfhe-development
git checkout v1.2.3
```
2) Build and install OpenFHE following the instructions in that repository. This will install the headers and libraries required by ELSA.

Build ELSA

After OpenFHE is installed and discoverable in your environment, clone and build ELSA as follows:

```
git clone https://github.com/nd-dsp-lab/
elsa_protocol.git
cd elsa_protocol
mkdir -p build && cd build
cmake -S .. -B .
make
```

[3]The OpenFHE library is available at https://github.com/openfheorg/openfhe-development.

If configuration and compilation succeed, multiple executables will be produced. Out of these executables, we are primarily concerned with these three:

- `main_psmt`: Runs the full ELSA pipeline, including encrypted label selection and analytics across distributed datasets. This corresponds to Section VIII-C that outlines the detailed results.
- `main_vaf`: Runs and evaluates only the VAF component, including weakDEP-based domain extension, iterative transformation, and cleanse steps. Appendix A goes into more detail about the parameters used in this executable (Table V) and how to tune them.
- `main`: Runs and evaluates the slotwise-windowing method for varying δ and κ values. Section VIII-B1 discusses the detailed results of running this executable with varying parameters.

C. Prepare Data

The directory `./data` in the root must contain all preprocessed datasets. The artifact expects these datasets to already be cleaned and formatted according to the instructions in `./data/README.md`. The `README` file describes how raw datasets are converted into the inputs consumed by the pipeline (for example, label domains, encodings, and partitioned shares for the distributed setting). Place the resulting files directly under `./data` before running any experiments.

D. Experiment Workflow

We describe how to execute the experiments in the same structure as in the paper. We separate the workflow into (a) running the full encrypted pipeline, (b) evaluating the VAF module, and (c) evaluating the slot-wise windowing method in isolation.

Full Pipeline (`main_psmt`)

1) **Parameter selection.** Decide the dataset location and encoding, then fix the command-line flags required by `main_psmt`:
   - `-DBPath <str>`: absolute or relative path to the directory that contains the dataset.
   - `-DBName <str>`: name of the dataset folder/file under `-DBPath`.
   - `-isSim <int>`: simulation toggle (decide to simulate the results from other servers or not. If 0, then it computes all the operations for all the participated servers. Default is 1).
   - `-isCompact <int>`: compact/space-efficient mode toggle (`0` or `1`; decide to use a compressed representation of the labels).
   - `-numChunks <int>`: set the number of ID ciphertexts held by each sender (horizontal segmentation of the database).

- `-itemLen <int>`: set the length of IDs (`1` or `2`). The actual length is `itemLen` $\times$ 64.
- `[-scalingModSize <int>]` (optional): scaling modulus size (max 59); set only if overriding defaults.

All flags above are required except `-scalingModSize`. Omitting any required flag will terminate execution with an error.

2) **Input preparation.** Ensure the directory given by `-DBPath` exists and contains the dataset named by `-DBName`. The encoding of items must match `-itemLen` (e.g., if `-itemLen` is 1, every item should be representable in 64 bits). A typical layout is `<DBPath>/<DBName>/....`
3) **Run the pipeline.** Execute `main_psmt` with the selected flags. For example:

```
./main_psmt
-DBPath ./data
-DBName VLDP
-isSim 0
-isCompact 1
-numChunks 8
-itemLen 1
-scalingModSize 42
```

This runs the protocol in non-simulation mode on the dataset `VLDP` found at `./data`, using compact mode, processing in 8 chunks with 64-bit items, and explicitly setting the scaling modulus size to 42.

4) **Collect outputs.** The binary reports correctness checks, timing for selection/aggregation, and parameter usage.

VAF-Only Evaluation (`main_vaf`)

1) **Select VAF parameters.** For a target domain size, collect parameters for the iterative VAF transformations from Table V.
2) **Run the VAF executable.** Execute:

```
./main_vaf [args...]
```

The VAF binary instantiates the wDEP base function (see paper for exact arguments), applies domain extension, and iterative transformations.

3) **Record the output.** The executable reports the exact behavior of the VAF and wDEP function in the homomorphic domain as specified in the paper.

Slot-Wise Windowing (`main`)

This executable takes δ and κ as positional arguments, automatically selects optimal VAF/wDEP presets and FHE depth for those values, runs the slot-wise windowing method, and prints a summary of results.

1) **Parameter selection.** Choose δ (so the domain size is 2^δ) and κ (the number of parts each item is parsed into).
2) **Run the pipeline.** Usage:

```
./main <delta> <kappa>
```

3) **Collect outputs.** The program will produce a correctness indicator for the proposed slotwise-windowing method and how it syncs with VAF and wDEP to produce a high-fidelity approximation of 1s and 0s.

Accuracy & Performance Measurements

The artifact should report that the encrypted analytics output matches the corresponding plaintext computation on the same data (up to around 24 bits). The artifact logs wall-clock runtime for the main stages, such as encrypted label selection and encrypted analytics. While absolute runtimes may vary across machines, the qualitative scaling trend should match the paper:

1) Runtime increases with domain size.
2) Required depth increases monotonically with domain size (see Table V).
3) The VAF construction with `newVAF` = T reduces the number of transformations needed compared to naive squaring, while staying within feasible FHE depth in OpenFHE.

Interpreting Table V

- The tuple $(k, L, R, n_{\mathsf{DEP}})$ defines the WeakDEP expansion schedule, where R is the base domain size before extension, L is the extension rate, and n_{DEP} is the number of DEP extensions applied.
- The pair $(n_{\mathsf{VAF}}, \texttt{newVAF})$ defines the iterative VAF transformation policy. If `newVAF` = F, the update rule is naive squaring $f(x) \mapsto f(x)^2$. If `newVAF` = T, we apply the optimized transformation that improves approximation quality without unnecessary multiplicative depth.
- The `depth` column gives the multiplicative depth budget required for homomorphic evaluation at the given domain size. This is the primary constraint for feasibility under OpenFHE.

Parameter Estimator: For custom parameter settings, we provide the parameter estimator written in Python, which is based on standard Python backend libraries such as `math`. This code implements the parameter selection methods specified in Appendix A. The estimator takes the values of k and L and the desired domain range, returning a set of VAF parameters that minimizes multiplication depth. For the target domain size M, the parameter presets in Table V can be reproduced by running the function $\texttt{DEPSelector}(k, L, M)$ that is implemented in the `.ipynb` file located under `./materials` directory. This function returns $R, n_{\mathsf{DEP}}, \epsilon_{\text{sep}}, n_{\mathsf{VAF}}, \texttt{newVAF}$, along with the total FHE depth and the supporting range.

Note that the function also allows custom parameters as long as the parameter choice is consistent with our analysis in Appendix A. For example, the code throws an error if $L^2 \geq K$ because our wDEP construction is undefined in this parameter regime.