electronics

MDPI

Article

Post-Quantum Private Set Intersection with Ultra-Efficient Online Performance

Yue Qin [1,2], Bei Liang [2,*], Hongyuan Cai [3] and Jintai Ding [4]

[1] School of Cyber Science and Technology, Beihang University, Beijing 100191, China
[2] Beijing Institute of Mathematical Sciences and Applications, Beijing 101408, China
[3] Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China
[4] School of Mathematics and Physics, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China
* Correspondence: lbei@bimsa.cn

Abstract

While tremendous progress has been made towards achieving highly efficient and practical Private Set Intersection (PSI) protocols during the last decade, the development of post-quantum PSI is still far from satisfactory. Existing post-quantum PSI protocols encounter a dilemma: while those based on fully homomorphic encryption (FHE) achieve low online communication, they suffer from significant online computation; conversely, protocols based on post-quantum Oblivious Pseudorandom Functions (OPRFs) exhibit excellent online computational performance but incur substantially high online communication. To overcome this dilemma, we present a lattice-based PSI protocol that achieves optimal online performance in both communication and computation. Our solution introduces two core innovations: a robust signal comparison algorithm based on RLWE key exchange, which determines the intersection through signal consistency rather than direct shared key comparison, and an optimized Oblivious Key–Value Stores (OKVS) implementation featuring a composite key–value mapping for efficient handling of high-dimensional RLWE polynomials. We implement the protocol and conduct extensive benchmarks in both symmetric and asymmetric set-size settings. The results show that our construction achieves the lowest online overhead in both computation and communication among all tests. For example, with asymmetric set sizes $(2^{12}, 11041)$, the online phase requires only 0.132 s, yielding 19× and 282× improvements over FHE-based (CCS'21) and OPRF-based (EUROCRYPT'25) protocols, respectively. Even at $(2^{24}, 11041)$, our online communication time is only 0.201 s, which is 226× and 184× that of FHE-based and OPRF-based PSI, respectively. Additionally, our online communication overhead is the lowest in all tests; however, this comes at the cost of heavy offline communication overhead for very large set sizes, revealing a clear trade-off between pre-computation and online efficiency. This work addresses a critical gap in post-quantum PSI by delivering a protocol that achieves balanced online communication and computational overhead, thereby enabling broader practical deployment.

Keywords: private set intersection; post-quantum key exchange; signal comparing algorithm

Check for updates

Academic Editor: Zbigniew Kotulski

Received: 14 November 2025
Revised: 13 December 2025
Accepted: 15 December 2025
Published: 19 December 2025

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Private Set Intersection (PSI) protocols are one of the commonly used two-party secure computation primitives, which enables two parties (i.e., a receiver and a sender), each of whom has a set (respectively, X and Y), to privately learn the intersection $X \cap Y$ without

disclosing additional information to the other party. This unique capability positions PSI as an ideal tool for facilitating data collaboration in privacy sensitive environments [1–3]. Over the years, PSI has found numerous applications in practice, including private contact discovery [4,5], online advertising [6,7], and privacy-preserving contact tracing for infectious diseases [8,9], among others. Due to its wide applications, tremendous progress has been made towards realizing PSI efficiently [10–15].

Given that the security of most widely used PSI protocols mostly relies on assumptions of classical number-theoretic hard problems, such as the discrete logarithm and integer factorization problems, which underpin the security of their core building blocks like Oblivious Transfer (OT) and Oblivious Pseudorandom Functions (OPRFs), it gives us confidence in the protocols' security nowadays. However, the rapid advancement of quantum computing technology has posed a significant challenge to the underlying hardness assumptions, and it is expected that Diffie–Hellman (DH) assumptions for elliptic curves can be broken by a quantum computer. This is not only a future threat since protocol transcripts collected today can be recorded and saved until a quantum computer is available, then decrypted in a Harvest Now, Decrypt Later attack [16,17]. Motivated by these concerns, alongside conventional security and performance considerations, the development and deployment of PSI protocols resilient to quantum attacks have become an essential research problem.

The problem of constructing post-quantum-based PSI protocols has been studied extensively in the literature. Debnath et al. [18] proposed a two-party Private Set Intersection cardinality (PSI-CA) protocol based on the hardness of the learning-with-errors problem, which achieves linear computational and communication complexity of the size of the inputs, i.e., $O(|X| + |Y|)$, where $|X|$ and $|Y|$ are the set sizes of the sender and receiver, respectively. Follow-up work [19] proposed a lattice-based multi-party PSI protocol under a star topology, achieving linear complexity of the input set size. Recently, Abudaqa et al. [20] presented instantiations of PSI from prominent post-quantum key encapsulation mechanisms, i.e., Crystals-Kyber and Classic-McEliece. While their Kyber-based PSI protocol which utilizes the multiplicative homomorphic property, requires 2681.22 KB of communication and 228.1 ms to compute an intersection on two sets with 700 elements each, their McEliece-based protocol that uses additive homomorphism in conjunction with a bloom filter data structure requires 1515.11 KB of communication and 19.70 ms of running time. As Abudaqa et al. [20] pointed out, the Kyber-based PSI will suffer from prohibitive computational and communication overhead when a large number of items are involved in the multiplication operation, which significantly limits its scalability in practice, while the McEliece-based PSI demonstrates better efficiency in performance but raises fundamental concerns regarding its security.

Chen et al. [2] proposed an efficient PSI protocol leveraging fully homomorphic encryption (FHE), which works particularly well when the receiver's set is significantly smaller than the sender's. It achieves communication complexity linear in the size of the smaller set, and logarithmic in the larger set. Chen et al. [2] strategically decoupled the protocol into a pre-processing phase that can be performed entirely offline without involving the counterpart and an efficient online phase. During pre-processing, the sender with the larger set locally performs computationally intensive tasks, such as hashing its set and pre-computing polynomial coefficients. During the online phase, the receiver encrypts each of its items y and sends ciphertexts to the sender. For each y, the sender then evaluates homomorphically the product of differences of y with all of the sender's items x, randomizes the product by multiplying it with a uniformly random non-zero plaintext, and sends the result back to the receiver. The result decrypts to zero precisely when y is in the sender's set, and to a uniformly random non-zero plaintext otherwise, revealing no information about the sender's set to the receiver. To intersect a large set of size $|X| = 2^{24}$

with a much smaller set $|Y| = 11041$, the FHE-based PSI [2] protocol requires an online running time of 44.7 s and a communication cost of 23.2 MB. Later, Chen et al. [21] used an Oblivious Pseudorandom Function (OPRF) pre-processing phase to significantly improve the performance of FHE-based PSI over [2], which achieves an online running time of 20.1 s and online communication cost of 41.48 MB when computing intersections on sets with size $|X| = 2^{24}$ and $|Y| = 11041$. The follow-up work of Cong et al. [22] goes a step further to promote the efficiency of FHE-based PSI, and yields a protocol with better computation cost, requiring only $\mathcal{O}(\sqrt{|X|})$ homomorphic multiplications, and communication complexity sublinear in the larger set size. For intersecting 2^{24} and 11041 item sets, their FHE-based PSI protocol reduces the online communication overhead by 71% over [21] (from 41.48 MB to 11.87 MB). Notice that an significant advantage of the FHE-based PSI is that it has zero communication overhead during the offline phase; however, its primary drawback lies in the substantial computational overhead and latency introduced by the intensive homomorphic operations required during the online phase, which remains a critical bottleneck for real-time applications.

Recently, Heimberger et al. [23] proposed a post-quantum OPRF protocol (called LEAP) based on a lattice-based PRF. By incorporating LEAP with the OPRF-based PSI framework proposed by Kiss et al. [24], Heimberger et al. [23] also achieved a post-quantum PSI protocol in the context of unbalanced settings, resulting in a highly efficient implementation. Their PSI protocol based on LEAP follows a two-phase structure. In the offline phase, the sender pre-processes its large set by computing PRF values and constructing a Cuckoo filter. During the online phase, the sender and receiver interact to evaluate the OPRF for each element in the receiver's set, after which the receiver performs a local filter lookup to identify the intersection. This architecture effectively offloads the majority of the computational load to the sender, thereby optimizing the receiver's online efficiency. For instance, with parameters targeting an intersection of sets of sizes $|X| = 2^{24}$ and $|Y| = 2^{15}$, the entire online phase takes approximately 90 s and incurs a total communication cost of about 722 MB. The outstanding online computation time of the LEAP-based PSI comes at the cost of high communication overhead.

Concurrently, Heimberger et al. [25] proposed an isogeny-based OPRF (called OPUS), and instantiated Kiss et al.'s PSI framework using OPUS to achieve a post-quantum PSI protocol. However, OPUS-based PSI has significantly higher computational and communication overhead compared to LEAP-based PSI. For instance, in a balanced PSI scenario with set sizes $|X| = |Y| = 2^{10}$, the computational latency of OPUS-based PSI becomes substantial, with the total online running time exceeding 16,000 s and the communication overhead increasing to approximately 53 MB.

The practical deployment of existing post-quantum PSI protocols is currently hindered by a performance gap where no protocol achieves high efficiency in both communication and computation. On one hand, FHE-based PSI protocols achieve low online communication, yet they suffer from significant online computation. On the other hand, post-quantum OPRF-based PSI protocols exhibit excellent online computational performance but incur substantially high online communication. Such a dilemma makes us ask following question:

Can we design a post-quantum PSI protocol that achieves optimal online communication and computational overhead simultaneously?

In this paper, we aim to design a post-quantum PSI protocol that achieves optimal online efficiency. Our primary objective is to minimize the online computation and communication overhead while ensuring robust security against quantum attacks. The target protocol is designed to deliver exceptional online performance across both balanced and unbalanced scenarios, handling set sizes ranging from small (2^7) to large (2^{24}). We antici-

pate that our work will not only bridge the current gap in high-performance post-quantum PSI design but also create new opportunities for the widespread application of privacy-preserving technologies in the quantum era.

1.1. Contributions

In this paper, we make positive progress towards addressing the above question by presenting a lattice-based PSI protocol that demonstrates ultra online performance. By shifting the majority of computational operations to the offline phase, we significantly minimize both online computation and communication overhead while maintaining efficient offline performance. In our proposed PSI protocol, we introduce a robust signal comparison algorithm ($Compare(\cdot)$). Unlike traditional Diffie–Hellman (DH)-based PSI, which compares shared keys, our PSI protocol determines the intersection by comparing signal consistency, while preserving the structural elegance of its DH-based counterparts. Furthermore, we optimize the Oblivious Key–Value Stores (OKVS) implementation from [26] with a dedicated composite key–value mapping for high-dimensional polynomials. This enhancement enables efficient polynomial encoding/decoding, facilitating high-performance storage and single-query retrieval of polynomials, which is crucial for our PSI protocol's efficiency.

Our PSI protocol delivers the lowest online running time among compared protocols, which only relies on the size of the receiver's set n_2. In the asymmetric setting, regardless of whether n_1 is small or large, for ours the online time remains between 0.063 and 0.116 s for $n_2 = 5535$. The results in the large-scale set size ($n_1 = 2^{24}$) scenario highlight this gap, at $n_2 = 5535$, where ours achieves speedups of 360× and 184× that of FHE-based [22] and post-quantum OPRF-based [23] PSI protocols, respectively.

Our PSI incurs the lowest and constant online communication which only depends on n_2. In the asymmetric setting, for ours the online communication is a constant 2.787 MB for $n_2 = 5535$. Compared with FHE-based and post-quantum OPRF-based PSI, the communication overhead is reduced by up to 3.12× and 44×, respectively.

The key reason of our protocol outperforming the state-of-the-art lies in the fact that the dominant communication overhead of our online phase is bounded by the receiver's set size n_2 which is significantly smaller than the sender's. As a result, the online communication is nearly independent of the sender's set size and remains close to constant. In addition, the signal comparison algorithm used to determine the intersection in the online phase only requires lightweight bitwise operations that scale with n_2, thereby enabling extremely low online computation for our post quantum PSI protocol.

Overall, our PSI has ultra online performance, with the lowest online time and online communication overhead among all the comparison experiments.

We note that the online performance gain is accompanied with higher offline communication costs. Nevertheless, for small and medium-size datasets (e.g., $n_1 \leq 2^{16}$) the total communication of our protocol remains the lowest. For large datasets (e.g., $n_1 \geq 2^{20}$), the offline communication is much heavier compared to the others.

1.2. Related Work

Existing research on PSI can be divided into two main categories. One is classical constructions, which use public keys and symmetric cryptography. The other is post-quantum constructions, which are instantiated based on assumptions related to Learning with Errors (LWE), Ring Learning with Errors (RLWE), or learning parity with noise (LPN).

1.2.1. Classical PSI

The design of the PSI protocol is highly dependent on the relationship between the set sizes of the two participating parties. Accordingly, its technical approach can be clearly divided into two main strands: balanced PSI and unbalanced PSI. There are

significant differences between the two in terms of design objectives, technical selections, and optimization strategies.

In balanced PSI, it is assumed that the sizes of the two parties' sets are on the same order of magnitude ($|X| \approx |Y| = n$). The core objective is to minimize the total overhead of the protocol execution (both computational cost and communication cost), that is, to seek the optimal asymptotic complexity with respect to n.

Public-key-based PSI. The PSI protocol based on the Diffie–Hellman key exchange was initially proposed by Meadows [27] and later refined for practical systems by Huberman et al. [28]. Its core idea is to exploit the commutativity property of DH key exchange, enabling intersecting elements to yield identical values after being encrypted by both parties. Freedman et al. [29] introduced a polynomial-based method, encoding sets as polynomial roots and using oblivious evaluation to find intersections. Kissner and Song [30] later generalized this with polynomial homomorphic encryption for broader set operations. Additionally, RSA blind signature technology can be utilized, where [31] transforms the comparison of sets into the comparison of blind signature results.

OT Extension and OPRF-based PSI. A pivotal advancement in practical balanced PSI came with the adoption of the Oblivious Transfer (OT) extension [10], which efficiently transforms a limited number of costly public-key operations into a large volume of low-cost symmetric-key operations. In the OPRF-PSI framework, the server computes tags for its elements using a secret key, while the client securely retrieves tags for its own elements via OT without disclosing them. Pinkas, Schneider, and Zohner [30] conducted a comprehensive analysis of OT extension-based PSI, comparing it favorably with alternative approaches and spurring its widespread adoption. The PSI protocol in [11] later marked a significant milestone by introducing an efficient batch OPRF construction, setting a new performance benchmark for balanced PSI with its remarkably low communication overhead and high computational efficiency. Further enhancements included permutation-based hashing by Pinkas, Schneider, and Zohner [32] to minimize collisions, along with subsequent refinements such as faster OPRF implementations [33] and additional optimizations in communication and computation costs via a simple-OT-based OPRF [34].

Circuit-based PSI. Huang et al. in 2012 [35] utilized Yao's garbled circuits to construct a sorting-comparison-shuffling circuit, enabling secure set intersection under the semi-honest model. However, this approach incurred substantial circuit size and communication overhead. Subsequently, Pinkas et al. [32] introduced permutation hashing techniques, significantly reducing computational costs. In 2018, they further adopted two-dimensional cuckoo hashing [36], bringing computational and communication overheads close to linear complexity. In 2019, by integrating Oblivious Programmable Pseudorandom Functions (OPPRF), they achieved the first circuit protocol with linear communication complexity [37]. Chandran et al. [38] further optimized the approach using batched OPPRFs, making progress in achieving linear computational and communication complexity.

In an unbalanced PSI scenario, the sizes of the two parties' sets vary greatly, requiring specialized protocol design to reduce the computational burden on the resource-constrained party.

OT/OPRF-based PSI. The work of [12] was groundbreaking, introducing the OT extension paradigm that shifted computational asymmetry to the server side, achieving linear complexity relative to the larger set size and enabling practical deployments where clients with small datasets could interact efficiently with servers holding large amounts of records. Later, OT/OPRF-based schemes incorporating hashing-to-bins emerged as the mainstream approach. This is attributed to their linear encoders and minimal public key cost on the critical path as demonstrated in studies such as [3,10,11,13]. Resende and Aranha [39] proposed a protocol in which the sender applies an Oblivious Pseudorandom

Function (OPRF) to the receiver's set Y. To reduce communication overhead, the sender compresses X' before transmitting it. However, this compression scales linearly with the size of the larger set. Kiss et al. [24] proposed another protocol following a similar framework, differing primarily in the implementation of the OPRF and the compression strategy. Resende and Aranha [39] uses a Diffie–Hellman-based OPRF, while Kiss et al. [24] uses oblivious evaluation of the AES function using garbled circuits. This modification makes local OPRF evaluation more efficient at the sender but at the expense of higher communication and computational costs at the receiver.

OKVS-based PSI. A significant advancement came with the utilization of Oblivious Key–Value Stores (OKVS). The foundational work of [40] demonstrated how a specific OKVS construction (PaXoS) could be leveraged to create fast PSI protocols, even in the malicious security model. This was subsequently amplified and formalized by [41], who provided a comprehensive study of OKVS and showed how they can be used to amplify PSI protocols, leading to improved communication complexity. The work of [42] pioneered this direction, constructing a fast OPRF and circuit-PSI from VOLE, which inherently utilizes an OKVS-like structure. This synergy was further optimized by [15], who achieved blazing-fast PSI rates by employing improved OKVS constructions and subfield Vector Oblivious Linear Evaluation (VOLE), marking a state-of-the-art performance.

1.2.2. Post-Quantum PSI

Post-quantum unbalanced PSI faces the challenge of maintaining the client efficiency advantages of classical approaches while incorporating quantum-resistant cryptographic primitives that typically involve larger parameters and more expensive operations.

FHE-based PSI. Chen et al. [2] proposed an unbalanced PSI protocol with low communication overhead by using RLWE-based leveled fully homomorphic encryption (FHE). In this work, the communication overhead is linear in the size of the (smaller) receiver's set and only logarithmic in the size of the (larger) sender's set. This foundational work was subsequently refined and extended in [21], which bolstered the security model through the integration of an OPRF, supported arbitrary-length items, and generalized the protocol to the Labeled PSI setting, enabling the transfer of arbitrary-length labels associated with the intersecting items. Building on this, an important recent advance is proposed in [22]. It achieves an asymptotically optimal computational cost of only $O(\sqrt{|X|})$ homomorphic multiplications and achieves sublinear communication complexity in larger sets $|X|$.

OPRF-based PSI. Constructions based on isogenies between supersingular elliptic curves represent an early line of research into post-quantum OPRFs. Boneh, Kogan, and Woo [43] proposed several isogeny-based OPRFs. However, an initial supersingular isogeny Diffie–Hellman (SIDH)-based construction was broken and later fixed within a framework of higher-dimensional isogenies. Another line of work uses commutative supersingular isogeny Diffie–Hellman (CSIDH) but often requires trusted setup or has high communication complexity [44]. A common challenge for isogeny-based OPRFs is their high computational overhead, with client requests taking seconds and server responses over a minute. This makes them generally unsuitable for PSI scenarios involving many OPRF evaluations.

Lattice assumptions are a promising foundation for practical post-quantum cryptography, leading to several OPRF proposals. The work of Heimberger et al. [23] introduces a highly efficient lattice-based OPRF based on the heuristic Ring-Learning with Rounding (RLWR) assumption. It builds upon the Spring PRF and uses OT and oblivious linear evaluation (OLE) for oblivious evaluation. Leap achieves online computation times of 750 microseconds for the server and 11 microseconds for the client, with online communication of just 23 kB. This efficiency profile makes Leap highly suitable for unbalanced PSI,

where a client with a smaller set performs many fast, low-communication OPRF evaluations against a server's large set.

Other recent advances on post-quantum OPRFs, such as LeOPaRd [45] and distributed OPRFs from Legendre PRFs [46], focus on verifiable or multi-server OPRF functionalities and do not directly yield two-party PSI protocols comparable to the schemes discussed above.

1.3. Road Map

The remainder of this paper is organized as follows. Section 2 introduces the notations and provides the necessary preliminaries. Subsequently, Section 3 details the construction of our proposed post-quantum PSI protocol, which achieves ultra-efficient online performance. It elaborates on the core components, with a particular focus on the implementation of the signal comparison algorithm. Section 4 presents the design of our optimized OKVS implementation. Section 5 focuses on the experimental evaluation, where we evaluate the performance, scalability, and overhead of our proposed solutions in comparison to state-of-the-art baselines across various realistic scenarios. Finally, Section 6 wraps up the paper by summarizing the key findings and discussing the broader implications of our work.

2. Preliminaries

2.1. Notations

Let n be a power of two, and let $q > 2$ be a prime modulus. Define the ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$. Each element in $\mathcal{R}_q$ is represented as a polynomial of degree less than n with coefficients in $\mathbb{Z}_q$. Let χ_α denote a discrete Gaussian distribution over $\mathbb{Z}^n$ with standard deviation α, which is extended to $\mathcal{R}_q$ by sampling each coefficient independently. All polynomials are in bold. λ denotes the cryptographic security parameter of the protocol.

2.2. Private Set Intersection

Private Set Intersection (PSI) enables two parties with private inputs to compute the intersection while revealing nothing beyond it. A protocol involves a sender $\mathcal{S}$ with input $X = \{x_1, \ldots, x_{n_1}\}$ and a receiver $\mathcal{R}$ with input $Y = \{y_1, \ldots, y_{n_2}\}$, where only $\mathcal{R}$ obtains $I = X \cap Y$. Security is formalized via the ideal functionality $\mathcal{F}_{\text{PSI}}$, which, on inputs X and Y, returns $X \cap Y$ to $\mathcal{R}$ and nothing to $\mathcal{S}$ (see Figure 1).

PARAMETERS:
- Sender's set X of size n_1
- Receiver's set Y of size n_2

Functionality:
- Receive X from $\mathcal{S}$ and Y from $\mathcal{R}$
- Compute $I = X \cap Y$ and output to $\mathcal{R}$

Figure 1. Ideal functionality $\mathcal{F}_{\text{PSI}}$.

A protocol Π is secure if its execution is indistinguishable from $\mathcal{F}_{\text{PSI}}$ under simulation.

Semi-Honest Security

Let $\text{view}_1^\Pi(X, Y)$ and $\text{view}_2^\Pi(X, Y)$ be the views of $\mathcal{S}$ and $\mathcal{R}$, and let $\text{output}^\Pi(X, Y)$ be $\mathcal{R}$'s output. Denote by $\mathcal{F}_{\text{PSI}}(X, Y)$ the ideal output.

Correctness. Π is correct if for all X, Y,

$$\Pr\left[\,\text{output}^\Pi(X, Y) \neq \mathcal{F}_{\text{PSI}}(X, Y)\,\right] \leq \text{negl}(\lambda). \tag{1}$$

Privacy. Π preserves privacy against semi-honest adversaries if there exist PPT simulators $\mathcal{S}_1, \mathcal{S}_2$ such that

$$\{ \mathcal{S}_1(1^\lambda, X) \} \overset{c}{\approx} \{ \mathsf{view}_1^\Pi(X, Y) \}, \tag{2}$$

$$\{ \mathcal{S}_2(1^\lambda, Y, \mathcal{F}_{\mathrm{PSI}}(X, Y)) \} \overset{c}{\approx} \{ \mathsf{view}_2^\Pi(X, Y) \}. \tag{3}$$

Thus, the sender's view can be simulated from X alone, and the receiver's view from Y and the intersection, ensuring that Π securely emulates $\mathcal{F}_{\mathrm{PSI}}$ in the semi-honest model.

2.3. RLWE-Based Key Exchange Protocol

We present an alternative description of the RLWE-based key exchange protocol introduced by Ding et al. [47], which enables two parties, denoted as P_1 and P_2, to establish a shared secret key over an insecure channel. The protocol relies on the hardness of the Ring Learning with Errors (RLWE) problem and employs a reconciliation mechanism to handle inherent noise.

2.3.1. RLWE Problems

The Ring Learning with Errors (RLWE) problem, introduced by Lyubashevsky et al. [48], is a structured variant of the Learning with Errors (LWE) problem. Let $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ with n being a power of two, and define $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ for an integer modulus $q \geq 2$.

The RLWE distribution $\mathcal{A}_{s,\chi}$ over $R_q \times R_q$ is parameterized by a secret $s \in R_q$, and is defined by sampling $a \leftarrow R_q$ uniformly and outputting $(a,\ b = a \cdot s + e)$, where $e \leftarrow \chi$ is an error polynomial drawn from a discrete Gaussian distribution.

The decision RLWE problem is to distinguish whether a sequence of samples is drawn from $\mathcal{A}_{s,\chi}$ or uniformly from $R_q \times R_q$. The RLWE problem is shown to be as hard as approximating the Shortest Vector Problem (SVP) on ideal lattices in the worst case, under quantum reductions.

The algebraic structure of R_q enables efficiency improvements, such as applying the Number Theoretic Transform (NTT) for fast polynomial multiplication. Variants of RLWE, such as the HNF-RLWE problem where the error is sampled from the entire ring, are also assumed to be hard.

2.3.2. RLWE-Based Post Quantum Key Exchange Protocol STEPS

Based on the hardness of the RLWE problem, the following protocol enables two parties to establish a shared secret key resistant to quantum adversaries. The protocol proceeds as follows:

1. Initialization: The public polynomial $\mathbf{a} \in \mathcal{R}_q$ is distributed to both parties.
2. P_1's initial message: P_1 samples secret polynomial $\mathbf{s}_1 \leftarrow \chi_\alpha$ and error polynomial $\mathbf{e}_1 \leftarrow \chi_\alpha$. Then P_1 computes $\mathbf{p}_1 = \mathbf{a} \cdot \mathbf{s}_1 + 2\mathbf{e}_1 \mod q$ and sends $\mathbf{p}_1$ to P_2.
3. P_2's response: P_2 samples secret polynomial $\mathbf{s}_2 \leftarrow \chi_\alpha$, error polynomials $\mathbf{e}_2 \leftarrow \chi_\alpha$, and $\mathbf{e}_2' \leftarrow \chi_\alpha$. And computes $\mathbf{p}_2 = \mathbf{a} \cdot \mathbf{s}_2 + 2\mathbf{e}_2 \mod q, \mathbf{k}_2 = \mathbf{p}_1 \cdot \mathbf{s}_2 + 2\mathbf{e}_2' \mod q$. After that, P_2 generates a signal $\sigma = Sig(\mathbf{k}_2)$, where Sig is the signal function (defined below). Computes the shared key $\mathrm{SK}_2 = \mathrm{Mod}_2(\mathbf{k}_2, \sigma)$, where Mod_2 is the robust extractor and sends $(\mathbf{p}_2, \sigma)$ to P_1.
4. P_1's key derivation: P_1 first samples $\mathbf{e}_1' \leftarrow \chi_\alpha$. Computes $\mathbf{k}_1 = \mathbf{s}_1 \cdot \mathbf{p}_2 + 2\mathbf{e}_1' \mod q$ and the shared key $\mathrm{SK}_1 = \mathrm{Mod}_2(\mathbf{k}_1, \sigma)$.

2.4. Signal and Extraction Functions

The signal function $Sig : \mathcal{R}_q \to \mathcal{R}_2$ is defined coefficient-wise. For a polynomial $\mathbf{x} = \sum_{i=0}^{n-1} x_i X^i \in \mathcal{R}_q$, the signal is computed as

$$Sig(\mathbf{x}) = \sum_{i=0}^{n-1} \sigma_b(x_i) X^i, \tag{4}$$

where $b \xleftarrow{\$} \{0,1\}$ is chosen uniformly at random for each invocation, and the functions $\sigma_0, \sigma_1 : \mathbb{Z}_q \to \{0,1\}$ are defined as

$$\sigma_0(x) = \begin{cases} 0 & \text{if } x \in [-\lfloor q/4 \rfloor, \lfloor q/4 \rfloor], \\ 1 & \text{otherwise,} \end{cases} \quad \sigma_1(x) = \begin{cases} 0 & \text{if } x \in [-\lfloor q/4 \rfloor + 1, \lfloor q/4 \rfloor + 1], \\ 1 & \text{otherwise.} \end{cases} \tag{5}$$

The robust extractor $\mathrm{Mod}_2 : \mathbb{Z}_q \times \{0,1\} \to \{0,1\}$ is defined for each coefficient as

$$\mathrm{Mod}_2(x, \omega) = \left(x + \omega \cdot \frac{q-1}{2} \mod q \right) \mod 2. \tag{6}$$

For a polynomial $\mathbf{x} \in \mathcal{R}_q$ and a signal polynomial $\omega \in \mathcal{R}_2$, the extraction is applied coefficient-wise.

2.4.1. Correctness

Under the condition $8(\alpha q)^2 n \leq \frac{q}{4} - 2$, the values $\mathbf{k}_1$ and $\mathbf{k}_2$ satisfy

$$\mathbf{k}_1 - \mathbf{k}_2 = 2\mathbf{e}_1 \cdot \mathbf{s}_2 + 2\mathbf{e}_1' - 2\mathbf{s}_1 \cdot \mathbf{e}_2 - 2\mathbf{e}_2' \mod (q, x^n + 1), \tag{7}$$

In [47] (Lemma 2), the magnitude of each coefficient in the difference is bounded by $\frac{q}{4} - 2$ with overwhelming probability. And Lemma 4 in [47] shows that if $8(\alpha q)^2 \cdot n \leqslant \frac{q}{4} - 2$, then $SK_1 = SK_2$ with overwhelming probability.

2.4.2. Security

The protocol is secure against passive adversaries under the RLWE assumption, which asserts that distinguishing between RLWE samples and uniform polynomials is computationally hard. The signal function does not leak information about the shared key due to the randomization in the choice of b, and the extracted key is uniformly random given the signal.

2.5. Oblivious Key–Value Stores (OKVS)

2.5.1. Definition of Oblivious Key–Value Stores (OKVS)

An OKVS enables encoding a set of n key–value pairs $(k_i, v_i)_{i=1}^n$ with $k_i \in \mathcal{K}$ and $v_i \in \mathcal{V}$ into an array $\mathbf{S} \in \mathcal{V}^m$ while preserving the confidentiality of the keys when the values are uniformly random. Formally, an OKVS scheme consists of two polynomial-time algorithms as follows.

The probabilistic encoding algorithm $\mathsf{Encode}(I; R) \to \mathbf{S}$ takes as inputs a set of n key–value pairs $I = (k_i, v_i)_{i=1}^n \in (\mathcal{K} \times \mathcal{V})^n$ with n distinct keys and randomness R, and outputs an encoding $\mathbf{S} \in \mathcal{V}^m \cup \{\perp\}$. The deterministic decoding algorithm $\mathsf{Decode}(\mathbf{S}, k; R) \to v$ takes as inputs the encoding $\mathbf{S} \in \mathcal{V}^m$ and randomness R, and outputs a value $v \in \mathcal{V}$.

The scheme must satisfy two crucial properties. Correctness requires that for all sets of n key–value pairs $I = (k_i, v_i)_{i=1}^n \subseteq (\mathcal{K} \times \mathcal{V})^n$ with n distinct keys, the probability $\Pr[\mathsf{Decode}(\mathbf{S}, k_i) \neq v_i \mid \mathbf{S} \leftarrow \mathsf{Encode}(I; R)]$ is bounded by ϵ for all $i \in [n]$, where the probability is taken over the choice of randomness R. Obliviousness guarantees that for any two sets of n distinct keys $A = \{k_1, \cdots, k_n\} \subseteq \mathcal{K}^n$ and $B = \{k_1', \cdots, k_n'\} \subseteq \mathcal{K}^n$ and values

$v_1, \ldots, v_n$ each drawn uniformly at random from $\mathcal{V}$, the distributions $\mathsf{Encode}(\{(k_i, v_i)\}_{i\in[n]})$ and $\mathsf{Encode}(\{(k'_i, v_i)\}_{i\in[n]})$ are computationally indistinguishable.

2.5.2. Construction of OKVS [26]

We recall the OKVS framework over a finite field $\mathbb{F}$ with key space $\mathcal{K}$. Given n distinct key–value pairs $\{(k_i, v_i)\}_{i=1}^n$, OKVS encodes them into an array $S \in \mathbb{F}^m$ of length $m = (1+\epsilon)n$, where $\epsilon > 0$ is a redundancy factor. Each key k determines a band of width w in a length-m row $r(k) \in \{0,1\}^m$: exactly w consecutive positions are non-zero and the rest are zeros. The band start index $a \in [m - w + 1]$ and the band contents are derived from independent hash functions (modeled as random oracles) H_1, H_2 seeded by public randomness. Stacking the rows $r(k_i)$ yields a random band matrix $M \in \{0,1\}^{n\times m}$. The encoder outputs any solution $S \in \mathbb{F}^m$ to $M \cdot S = [v_1, \ldots, v_n]^\top \in \mathbb{F}^n$. Decoding a query key k is the dot product $r(k) \cdot S$, which touches only the w-cell band. For parameters $m = (1+\epsilon)n$, $w = \Theta\left(\frac{\lambda_{\mathrm{OKVS}}}{\epsilon} + \log n\right)$, the matrix M has full row rank except with probability at most $2^{-\lambda_{\mathrm{OKVS}}}$ to ensure correctness. Encoding runs in $O(nw)$ word operations using band-aware Gaussian elimination, while decoding takes $O(w)$. We denote the resulting encode and decode algorithms by $\mathsf{Enc}_{\mathrm{RB}}$ and $\mathsf{Dec}_{\mathrm{RB}}$, respectively, and give them in Algorithms 1 and 2.

Algorithm 1: $\mathsf{Enc}_{\mathrm{RB}}$ (OKVS Encode)

Input : Distinct pairs $\{(k_i, v_i)\}_{i=1}^n$; redundancy $\epsilon > 0$; band width w; public randomness $R = (R_1, R_2)$; field $\mathbb{F}$.
Output: Encoding array $S \in \mathbb{F}^m$ with $m = (1+\epsilon)n$.

$m \leftarrow (1+\epsilon)n$; initialize sparse/banded matrix $M \in \{0,1\}^{n\times m}$
for $i \leftarrow 1$ **to** n **do**
  $a_i \leftarrow H_1(R_1, k_i) \in [1, m - w + 1]$
  $u_i \leftarrow H_2(R_2, k_i) \in \{0,1\}^w$ **for** $j \leftarrow 1$ **to** w **do**
    $M[i,\, a_i + j - 1] \leftarrow u_i[j]$
  end
end
Sort rows of M by band start a_i (stable)
Solve $M \cdot S = [v_1, \ldots, v_n]^\top$ using band-aware Gaussian elimination with back-substitution
if *system has free variables* **then**
  Assign free entries of S uniformly at random in $\mathbb{F}$
end
return S

Algorithm 2: $\mathsf{Dec}_{\mathrm{RB}}$ (OKVS Decode)

Input : Encoding $S \in \mathbb{F}^m$; query key $k \in \mathcal{K}$; band width w; randomness $R = (R_1, R_2)$.
Output: Decoded value $v \in \mathbb{F}$.

$a \leftarrow H_1(R_1, k) \in [1, m - w + 1]$
$u \leftarrow H_2(R_2, k) \in \{0,1\}^w$
$v \leftarrow 0$
for $j \leftarrow 1$ **to** w **do**
  $v \leftarrow v + u[j] \cdot S[a + j - 1]$
end
return v

3. Our Post-Quantum PSI Protocol

3.1. Construction

Our post-quantum PSI protocol, called signal-based PSI $\Pi_{\text{Sig-PSI}}$, is constructed from the RLWE-based Ding Key Exchange scheme. The design goal is to enable two parties, referred to as the sender holding a set $X = \{x_i\}_{i\in[n_1]}$ and the receiver holding a set $Y = \{y_j\}_{j\in[n_2]}$, to compute the intersection $X \cap Y$ while revealing no additional information. The novelty of our approach lies in leveraging signal comparison rather than shared key extraction, combining with the usage of OKVS Encode/Decode technology to reduce the computation and communication overhead.

Our protocol is presented in Figure 2, which proceeds with two phases, an offline phase and an online phase. The offline phase begins with the sender sampling its secret key $\mathbf{s}_1$ and error terms $\mathbf{e}_{1,i} \leftarrow \chi_\alpha$ for $i \in [n_1]$. For each input element x_i, the sender computes a public key $\mathbf{P}_{1,i} = H(x_i)\mathbf{s}_1 + 2\mathbf{e}_{1,i} \pmod q$. These public keys are then embedded into an OKVS, resulting in an encoded structure $S = \textbf{OKVS.Encode}(\{(h(x_i), \mathbf{P}_{1,i})\}_{i\in[n_1]})$, which is transmitted to the receiver.

Parameters:
– χ_α: discrete Gaussian distribution with standard deviation α
– $\mathbb{Z}_q$: the ring with all coefficients are integers modulo q
– $\mathcal{R}_q$: polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$
– $H(\cdot)$: $\{0,1\}^* \to \mathcal{R}_q$: the hash function which can be modeled as a random oracle

Sender		**Receiver**
Input : $X, \lvert X\rvert = n_1$		Input: $Y, \lvert Y\rvert = n_2$
1. Sample $\mathbf{s}_1, \mathbf{e}_{1,i} \leftarrow \chi_\alpha$, for $i \in [n_1]$		
2. For each $i \in [n_1]$:		
- $\mathbf{P}_{1,i} = H(x_i)\mathbf{s}_1 + 2\mathbf{e}_{1,i} \pmod q$		
3. $S = \textbf{OKVS.Encode}(\{(h(x_i), \mathbf{P}_{1,i})\}_{i\in[n_1]})$	$\xrightarrow{S}$	
		4. Sample $\mathbf{s}_2, \mathbf{e}_{2,j} \leftarrow \chi_\alpha$, for $j \in [n_2]$
		5. For each $j \in [n_2]$:
		- $\mathbf{P}_{2,j} = H(y_j)\mathbf{s}_2 + 2\mathbf{e}_{2,j} \pmod q$
	$\xleftarrow{\{\mathbf{P}_{2,j}\}_{j\in[n_2]}}$	
6. For each $j \in [n_2]$:		
- $\mathbf{K}_{1,j} = \mathbf{P}_{2,j}\mathbf{s}_1 + 2\mathbf{g}_{1,j} \pmod q$, $\mathbf{g}_{1,j} \leftarrow \chi_\alpha$		
- $\omega_{1,j} = Sig(\mathbf{K}_{1,j})$ (shuffled)	$\xrightarrow{\omega_1=\{\omega_{1,j}\}_{j\in[n_2]}}$	
		7. For each $j \in [n_2]$:
		- $\mathbf{P}'_1(y_j) \leftarrow \textbf{OKVS.Decode}(S, h(y_j))$
		- $\mathbf{K}_{2,j} = \mathbf{P}'_1(y_j)\mathbf{s}_2 + 2\mathbf{g}_{2,j} \pmod q$, $\mathbf{g}_{2,j} \leftarrow \chi_\alpha$
		- $\omega_{2,j} = Sig(K_{2,j})$
		8. Let $\omega_2 = \{\omega_{2,j}\}_{j\in[n_2]}$ and $\mathbf{K}_2 = \{\mathbf{K}_{2,j}\}_{j\in[n_2]}$
		Output $I = Compare(\omega_1, \omega_2, \mathbf{K}_2)$

Figure 2. Our Post-Quantum PSI $\Pi_{\text{Sig-PSI}}$.

Upon receiving S, the online phase begins. The receiver samples its own secret key $\mathbf{s}_2$ and error terms $\mathbf{e}_{2,j}$ for $j \in [n_2]$, and computes its corresponding public keys $\mathbf{P}_{2,j} = H(y_j)\mathbf{s}_2 + 2\mathbf{e}_{2,j} \pmod q$. These public keys are returned to the sender. After receiving $\{\mathbf{P}_{2,j}\}_{j\in[n_2]}$, the sender derives intermediate auxiliary values $\mathbf{K}_{1,j} = \mathbf{P}_{2,j}\mathbf{s}_1 + 2\mathbf{g}_{1,j} \pmod q$, where $\mathbf{g}_{1,j} \leftarrow \chi_\alpha$ are fresh error terms. Each $\mathbf{K}_{1,j}$ is then mapped to a signal $\omega_{1,j} = Sig(\mathbf{K}_{1,j})$, and the set $\omega_1 = \{\omega_{1,j}\}_{j\in[n_2]}$ is transmitted to the receiver. In parallel, the receiver decodes the OKVS on its input $h(y_j)$ to obtain $\mathbf{P}'_{1,j} = \textbf{OKVS.Decode}(S, h(y_j))$, and computes its own intermediate auxiliary values $\mathbf{K}_{2,j} = \mathbf{P}'_{1,j}\mathbf{s}_2 + 2\mathbf{g}_{2,j} \pmod q$ with $\mathbf{g}_{2,j} \leftarrow \chi_\alpha$. The receiver then computes signals $\omega_2 = \{\omega_{2,j}\}_{j\in[n_2]} = Sig(\mathbf{K}_{2,j})$ for each input element.

Finally, the intersection is obtained through the signal matching procedure. The receiver executes the algorithm $Compare(\omega_1, \omega_2, n_2)$, which determines whether signals generated by the two parties correspond to the same input element. The correctness of this comparison follows from the property that if $x_i = y_j$, then the difference $\mathbf{K}_{1,j} - \mathbf{K}_{2,j}$ reduces to a small noise term with overwhelming probability. As a consequence, the signals

$\omega_{1,j}$ and $\omega_{2,j}$ will be equal with high probability. This guarantees that the receiver outputs $I = X \cap Y$ correctly, while ensuring that no information about non-intersecting elements is revealed.

3.2. Implementation of the $Compare(\cdot)$ Algorithm

The $Compare(\cdot)$ algorithm operates by analyzing two signal sets w_1 and w_2, each of which consists of n_2 signals, corresponding to two private sets X and Y to determine their intersection I. The correctness of $Compare(\cdot)$ algorithm is ensured by the fact that a signal match occurs if and only if the corresponding elements from the two private sets are equal. In order to efficiently implement $Compare(\cdot)$ between two sets of signals, a sub-algorithm for a one-to-one signal matching $Match(\cdot)$ is provided firstly, which takes as inputs signals $w_{1,i}$ and $w_{2,j}$ from each set of w_1 and w_2 and returns a bit to indicate if two signals match or not, i.e., 0 (mismatch) or 1 (match). The $Compare(\cdot)$ algorithm invokes the $Match(\cdot)$ subroutine $(n_2)^2$ times to identify all matching signal pairs, thereby allowing the receiver to output the intersection $X \cap Y = \{y_j \mid \exists i \in [n_2] \text{ s.t. } Match(w_{1,i}, w_{2,j}) = 1 \text{ for } j \in [n_2]\}$.

The $Match(\cdot)$ sub-algorithm specifically operates on a pair of input signals $w_{1,i} \in w_1$ and $w_{2,j} \in w_2$, which are derived from the ring elements $\mathbf{K}_{1,i}$ and $\mathbf{K}_{2,j}$ respectively via the signal function $Sig(\cdot)$. Thus, we can determine the relationship between $\mathbf{K}_{1,i}$ and $\mathbf{K}_{2,j}$ by comparing whether the signals match, and further infer whether $y_j \in X$. If $y_j = x_i \in X$, the signal values corresponding to the ring elements $\mathbf{K}_{1,i}$ and $\mathbf{K}_{2,j}$ obtained from y_j are consistent, and $Match(\cdot)$ outputs 1; otherwise it outputs 0. According to the definition of the signal function Equation (4), when $\mathbf{K}_{1,i}$ or $\mathbf{K}_{2,j}$ lies within the interval $(-\frac{q}{2}, -\frac{q}{4}) \cup (\frac{q}{4}, \frac{q}{2})$ the output value is 1, and a value of 0 when it is within $[-\lfloor q/4 \rfloor, \lfloor q/4 \rfloor]$. Intuitively, these two cases correspond to the blue and white segments, respectively, as illustrated in the top bar of the Figure 3. We denote $\pm\tau = \pm\lfloor q/4 \rfloor$ as the signal flips point, which means the signals may be different if $\mathbf{K}_{1,i}$ and $\mathbf{K}_{2,j}$ are around $\pm\tau$. The errors upper bound is denoted as η. Accordingly, three regions are delineated as illustrated in Figure 3:

- certain-1 region (orange segments): $(-\frac{q}{2}, -\tau - \eta) \cup (\tau + \eta, \frac{q}{2})$. If $\mathbf{K}_{2,j}$ resides in this region, and $|\mathbf{K}_{2,j} - \mathbf{K}_{1,i}| \leq \eta$, then it has $Sig(\mathbf{K}_{1,i}) = Sig(\mathbf{K}_{2,j}) = 1$.
- certain-0 region (white segment): $(-\tau + \eta, \tau - \eta)$. If $\mathbf{K}_{2,j}$ is located in this region, and $|\mathbf{K}_{2,j} - \mathbf{K}_{1,i}| \leq \eta$, then it has $Sig(\mathbf{K}_{1,i}) = Sig(\mathbf{K}_{2,j}) = 0$.
- uncertain region (shaded segment): $(-\tau - \eta, -\tau + \eta) \cup (\tau - \eta, \tau + \eta)$. If $\mathbf{K}_{2,j}$ falls within this region, signal flipping may occur due to errors. However, since this uncertain region is very small (with $q \approx 2^{20}$ significantly larger than the error upper bound η), its impact on matching correctness is negligible.

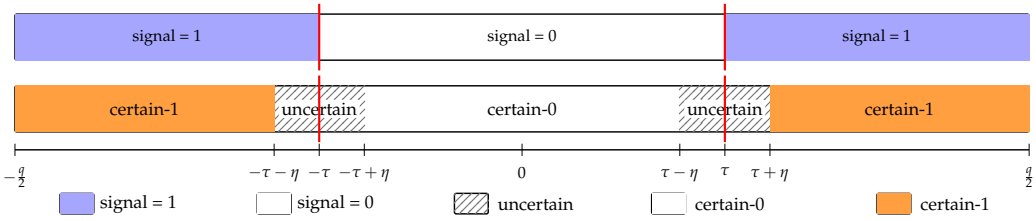


Figure 3. Top bar: partition induced by $Sig(\cdot)$ on the centered axis $[-\frac{q}{2}, \frac{q}{2}]$ (blue: $|u| \geq \tau \Rightarrow$ signal $= 1$). Bottom bar: orange represents the certain-1 region ($|u| \geq \tau + \eta$), white represents the certain-0 region ($|u| \leq \tau - \eta$), hatched represents uncertain around $\pm\tau$. Vertical red lines mark the signal flips point at $\pm\tau$ with $\tau = \lfloor q/4 \rfloor$.

In our PSI protocol, if the receiver's element $y_j \notin X$, the value $\mathbf{P}'_1(y_j)$ obtained from the decoding of OKVS on y_j is random, then the corresponding $\mathbf{K}_{2,j}$ is also random. Therefore, the probability that two signals $w_{1,i}$ and $w_{2,j}$ match is negligible, i.e., $Match(w_{1,i}, w_{2,j}) = 0$ with overwhelming probability. If $y_j = x_i \in X$, then $H(y_j) = H(x_i)$.

We have OKVS.Decode($H(y_j)$) = $\mathbf{P}'_1(y_j) = \mathbf{P}_{1,i} = H(x_i)\mathbf{s}_1 + 2\mathbf{e}_{1,i} \pmod q$. The corresponding signals are

$$\omega_{1,i} = \mathrm{Sig}(\mathbf{K}_{1,i}) = \mathrm{Sig}((H(y_j)\mathbf{s}_1\mathbf{s}_2 + 2(\mathbf{e}_{2,j}\mathbf{s}_1 + \mathbf{g}_{1,j})) \bmod q),$$
$$\omega_{2,j} = \mathrm{Sig}(\mathbf{K}_{2,j}) = \mathrm{Sig}((H(y_j)\mathbf{s}_1\mathbf{s}_2 + 2(\mathbf{e}_{1,i}\mathbf{s}_2 + \mathbf{g}_{2,j})) \bmod q).$$

Since $\mathbf{K}_{2,j} - \mathbf{K}_{1,i} = 2(\mathbf{e}_{1,i}\mathbf{s}_2 + \mathbf{g}_{2,j} - \mathbf{e}_{2,j}\mathbf{s}_1 - \mathbf{g}_{1,j})$, given that the upper bound of the error term is η, then we have that $Sig(\mathbf{K}_{1,i}) = Sig(\mathbf{K}_{2,j}) = 0$ if $K_{2,j}$ is in the certain-0 region, and $Sig(\mathbf{K}_{1,i}) = Sig(\mathbf{K}_{2,j}) = 1$ if $K_{2,j}$ is in the certain-1 region. Owing to the large value of q, certain regions take up approximately 99% of the domain. If $\mathbf{K}_{2,j}$ lies in the uncertain region, e.g., $(-\tau, -\tau + \eta)$, then $Sig(\mathbf{K}_{2,j}) = 0$. Otherwise, it may happen that $\mathbf{K}_{1,i}$ lies in $(-\tau - \eta, -\tau)$, which implies $Sig(\mathbf{K}_{1,i}) = 1$, leading to inconsistency between two signals. Given that the range of the uncertainty region is extremely small, in this case, $Match(\cdot)$ determines whether two signals match by counting the proportion of consistent signal bits between the two signals and setting a threshold (e.g., 99%) to ensure high accuracy.

3.2.1. $Match(\cdot)$ Algorithm

All operations are in $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$. We use the unique centered set of residues $\mathcal{D}_q = \left(-\frac{q}{2}, \frac{q}{2}\right] \cap \mathbb{Z}$, and henceforth assume all coefficients are represented in $\mathcal{D}_q$.

Under the assumptions of Theorem 2 (random oracle for H, i.i.d. discrete-Gaussian secrets and errors), the per-coefficient difference noise between $\mathbf{K}_{1,j}$ and $\mathbf{K}_{2,j}$ is subgaussian with proxy

$$\alpha_\xi = \sqrt{8n\alpha^4 + 8\alpha^4}, \qquad h_1 = 5\,\alpha_\xi. \tag{8}$$

To reason on the integer lattice, we use the discrete margin $\eta = \lceil h_1 \rceil \in \mathbb{N}$.

A centered ring element $u \in \mathcal{D}_q$ is in the certain region $C^{\mathbb{Z}}_{\tau,\eta}$ (or uncertain region $U^{\mathbb{Z}}_{\tau,\eta}$) if it is at a distance of at least η from the decision boundaries $\pm\tau$:

$$C^{\mathbb{Z}}_{\tau,\eta} = \{|u| \le \tau - \eta\} \cup \{|u| \ge \tau + \eta\},\ U^{\mathbb{Z}}_{\tau,\eta} = \{\big||u| - \tau\big| < \eta\}. \tag{9}$$

The procedure of $Match(\cdot)$ is shown in Algorithm 3, which compares two signals $\omega_{1,i}, \omega_{2,j} \in \{0,1\}^\ell$ derived from $\mathbf{K}_{1,i}, \mathbf{K}_{2,j}$ using $\mathbf{K}_{2,j}$ as an object of reference. For $t = \{1, \ldots, \ell\}$, two steps are performed: (i) if the coefficient u_t (the t-th coefficient of $\mathbf{K}_{2,j}$) lies in $C^{\mathbb{Z}}_{\tau,\eta}$ and the signal bits are unequal, it immediately rejects; (ii) whenever the signals bits are consistent, regardless of whether u_t is in the certain region or the uncertain region, a consistency counter *Count* is incremented. The two signals are considered equal if and only if the final consistency count reaches the threshold $\psi\ell$.

3.2.2. $Compare(\cdot)$ Algorithm

Algorithm 4 shows the procedure of $Compare(\cdot)$. Given two sets of signals $\omega_1 = \{\omega_{1,1}, \ldots, \omega_{1,n_2}\}$ and $\omega_2 = \{\omega_{2,1}, \ldots, \omega_{2,n_2}\}$, for each pair of signals $\omega_{1,i}, \omega_{2,j}$ where $i, j \in [n_2]$, it invokes the $Match(\omega_{1,i}, \omega_{2,j}; \mathbf{K}_{2,j}, \psi)$ (Algorithm 3) to determine whether any two signals match or not. For each $j = 1, \ldots, n_2$, if there exists $i \in [n_2]$ such that $Match(\omega_{1,i}, \omega_{2,j}; \mathbf{K}_{2,j}, \psi) = 1$, then we include y_j in the intersection set.

Algorithm 3: Match $(\omega_{1,i}, \omega_{2,j};\ \mathbf{K}_{2,j},\ \psi)$

Input : Signals $\omega_{1,i}, \omega_{2,j} \in \{0,1\}^{\ell}$; ring element $\mathbf{K}_{2,j} \in \mathcal{R}_q$; threshold $\psi \in (0,1)$
Output: $\gamma \in \{0,1\}$ (1 = match, 0 = mismatch)

Conventions: $\tau = \lfloor q/4 \rfloor$, $\eta = \lceil h_1 \rceil$, $\langle \cdot \rangle_q \in \mathcal{D}_q$.
$C^{\mathbb{Z}}_{\tau,\eta} = \{u \in \mathcal{D}_q : |u| \le \tau - \eta\} \cup \{u \in \mathcal{D}_q : |u| \ge \tau + \eta\}$
$\gamma \leftarrow 1$; $Count \leftarrow 0$
for $t \leftarrow 1$ **to** ℓ **do**
  $u \leftarrow \langle \mathbf{K}_{2,j}[t] \rangle_q$
  if $\left(u \in C^{\mathbb{Z}}_{\tau,\eta}\right)$ **then**
    if $\omega_{1,i}[t] \neq \omega_{2,j}[t]$ **then**
      $\gamma \leftarrow 0$; **break**
    else
      $Count \leftarrow Count + 1$
    end
  else
    if $\omega_{1,i}[t] = \omega_{2,j}[t]$ **then**
      $Count \leftarrow Count + 1$
    end
  end
end
if $Count < \psi \cdot \ell$ **then**
  $\gamma \leftarrow 0$
end
return γ

Algorithm 4: Compare$(\omega_1, \omega_2, \mathbf{K}_2)$

Input : Two sets of signals $\omega_1 = \{\omega_{1,1}, \ldots, \omega_{1,n_2}\}$ and $\omega_2 = \{\omega_{2,1}, \ldots, \omega_{2,n_2}\}$.
$\mathbf{K}_2 = \{\mathbf{K}_{2,j}\}_{j \in [n_2]}$
Output: Intersection set I

$I \leftarrow \emptyset$
for $i \leftarrow 1$ **to** n_2 **do**
  for $j \leftarrow 1$ **to** n_2 **do**
    if $Match(\omega_{1,i}, \omega_{2,j}, \mathbf{K}_{2,j}, \psi) = 1$ **then**
      $I \leftarrow I \cup \{\omega_{2,j}\}$; **break**
  end
end
return I

3.2.3. Completeness and Soundness

Assumption 1 (Errors tail per coefficient). *For each coefficient t, $\Pr[|\xi[t]| > h_1] \le 2^{-21.73}$, where $h_1 = 5\alpha_\xi$ and $\alpha_\xi = \sqrt{8n\alpha^4 + 8\alpha^4}$.*

Assumption 2 (Errors bound). *With overwhelming probability, $\|\xi\|_\infty \le h_1$ for the accumulated noise $\xi = 2(\mathbf{e}_{2,j}\mathbf{s}_1 + \mathbf{g}_{1,j} - \mathbf{e}_{1,i}\mathbf{s}_2 - \mathbf{g}_{2,j})$.*

Lemma 1 (Coefficient-wise Signal Consistency under Bounded Perturbation)**.** *Under Assumptions 1 and 2, for any coefficient position* $t \in \{1, \ldots, \ell\}$*, let* $u = \langle \mathbf{K}_{2,j}[t] \rangle_q$ *and* $v = \langle \mathbf{K}_{1,i}[t] \rangle_q = u + \xi[t] \mod q$ *with* $|\xi[t]| \leq h_1$*. Then we have the following:*

1. *If* $u \in C^Z_{\tau,\eta}$*, then* $\omega_{1,i}[t] = \omega_{2,j}[t]$ *with probability 1.*
2. *If* $u \notin C^Z_{\tau,\eta}$*, then* $\omega_{1,i}[t] = \omega_{2,j}[t]$ *with probability at least* $\frac{1}{2}$*.*

Proof of Lemma 1. We consider each case separately.

1. $u \in C^Z_{\tau,\eta}$

We have two cases based on the definition of $C^Z_{\tau,\eta}$:

(1) $|u| \leq \tau - \eta$:

Since $|\xi[t]| \leq h_1$ and $\eta = [h_1]$, we have $h_1 \leq \eta + 1$. Therefore,

$$|v| = |u + \xi[t]| \leq |u| + |\xi[t]| \leq (\tau - \eta) + h_1 \leq \tau$$

Thus, both u and v fall within the interval $[-\tau, \tau]$ where $\sigma_0(x) = 0$.

For σ_1, the interval for 0 is $[-\tau + 1, \tau + 1]$. Since $|u| \leq \tau - \eta \leq \tau - 1$ (as $\eta \geq 1$) and $|v| \leq \tau$, both values are within this interval, so $\sigma_1(u) = \sigma_1(v) = 0$.

Therefore, regardless of the random bit b chosen for the signal function, $\omega_{1,i}[t] = \omega_{2,j}[t]$.

(2) $|u| \geq \tau + \eta$:

We have

$$|v| = |u + \xi[t]| \geq |u| - |\xi[t]| \geq (\tau + \eta) - h_1 \geq \tau$$

since $h_1 \leq \eta + 1$.

For σ_0: If $|v| > \tau$, then $\sigma_0(v) = 1 = \sigma_0(u)$. The edge case $|v| = \tau$ occurs only when $\xi[t] = \tau - u$, but since $|u| \geq \tau + \eta$, we have $|\xi[t]| \geq \eta$. Given that $|\xi[t]| \leq h_1$ and $h_1 = 5\alpha_\xi$ is chosen to make large deviations exponentially unlikely, this edge case occurs with negligible probability.

For σ_1: The interval for 0 is $[-\tau + 1, \tau + 1]$. Since $|u| \geq \tau + \eta \geq \tau + 1$ (as $\eta \geq 1$), we have $\sigma_1(u) = 1$. Similarly, $|v| \geq \tau$, and if $|v| \geq \tau + 1$, then $\sigma_1(v) = 1$. The edge case $|v| = \tau$ requires $|\xi[t]| = \eta$, which again occurs with negligible probability.

Therefore, with overwhelming probability, $\omega_{1,i}[t] = \omega_{2,j}[t]$ in this case.

2. $u \notin C^Z_{\tau,\eta}$

Refer to Figure 3. For any coefficient u in the uncertain region (hatched area where $\tau - \eta < |u| < \tau + \eta$), we prove that $\Pr[\omega_{1,i}[t] = \omega_{2,j}[t]] \geq \frac{1}{2}$.

Focus on a fixed u in the uncertain (hatched) region. Partition the bounded noise values d (where $|d| \leq h_1$) into two sets:

$$A = \{d : \sigma_b(u) = \sigma_b(u + d)\} \quad \text{(Agreement)}$$
$$B = \{d : \sigma_b(u) \neq \sigma_b(u + d)\} \quad \text{(Disagreement)}$$

Set A contains all noise values d for which the signal remains consistent, meaning $\sigma_b(u) = \sigma_b(u + d)$. Conversely, set B contains all noise values d that cause a disagreement, meaning $\sigma_b(u) \neq \sigma_b(u + d)$.

The proof hinges on constructing a symmetry mapping $f : B \to A$ defined simply by $f(d) = -d$. The geometric intuition from Figure 3 clarifies why this mapping works. Consider a point u in the right hatched region. A positive noise term d might be just large enough to push $u + d$ across the red threshold line at τ into the blue region, thereby flipping the signal and causing a disagreement. However, the symmetric negative noise term $-d$ would pull u further left, away from the threshold and deeper into the white region. This movement ensures that both u and $u - d$ remain on the same side of the threshold, thus preserving signal agreement. An analogous situation occurs for a point u in the left hatched region.

Crucially, the symmetry of the noise distribution guarantees that $\Pr[\xi = d] = \Pr[\xi = -d]$. Furthermore, the symmetric nature of the signal function's decision boundaries, mirrored around zero, ensures that the mapping f is a probability-preserving bijection from the disagreement set B into the agreement set A. This directly implies that the total probability mass of B cannot exceed that of A. Formally, $p_{\text{agree}} = \sum_{d \in A} \Pr[\xi = d] \geq \sum_{d \in B} \Pr[\xi = d] = p_{\text{disagree}}$.

Given that the probabilities of agreement and disagreement must sum to 1, the inequality $p_{\text{agree}} \geq p_{\text{disagree}}$ necessarily leads to the conclusion that $p_{\text{agree}} \geq \frac{1}{2}$.

This completes the proof of Lemma 1. □

Theorem 1 (Correctness of Algorithm 3)**.** *Assume the RLWE problem is hard, and parameters are chosen such that* $\Pr[\|\xi\|_\infty > h_1]$ *is negligible. Then we have the following:*

1. *Completeness: For matching keys with* $\mathbf{K}_{1,i} = \mathbf{K}_{2,j} + \xi \mod q$ *and* $\|\xi\|_\infty \leq h_1$*, Algorithm 5 outputs* $\gamma = 1$ *with probability at least* $1 - \epsilon_1$*, where* ϵ_1 *is negligible in the security parameter* λ*.*
2. *Soundness: For non-matching keys or keys with excessive noise (*$\|\xi\|_\infty > h_1$*), Algorithm 3 outputs* $\gamma = 0$ *with probability at least* $1 - \epsilon_2$*, where* ϵ_2 *is negligible in* λ*.*

Proof of Theorem 1. We prove each property separately.

1. Completeness:

Let X_t be the indicator random variable for the event that $\omega_{1,i}[t] = \omega_{2,j}[t]$ at position t. From Lemma 1, we have the following:

- If $u \in C^Z_{\tau,\eta}$ (which occurs with probability p_C), then $\Pr[X_t = 1] = 1$.
- If $u \notin C^Z_{\tau,\eta}$ (which occurs with probability $1 - p_C$), then $\Pr[X_t = 1] \geq \frac{1}{2}$.

The expected number of agreeing positions is

$$\mathbb{E}[\text{Count}] = \sum_{t=1}^{\ell} \mathbb{E}[X_t] \geq \ell \cdot p_C \cdot 1 + \ell \cdot (1 - p_C) \cdot \frac{1}{2} = \ell \cdot \left(\frac{1 + p_C}{2} \right)$$

Using the Chernoff bound, for any $\delta > 0$:

$$\Pr[\text{Count} < (1 - \delta) \cdot \mathbb{E}[\text{Count}]] \leq \exp\left(-\frac{\delta^2 \cdot \mathbb{E}[\text{Count}]}{2} \right)$$

Setting $\psi = (1 - \delta) \cdot \frac{1 + p_C}{2}$, we have

$$\Pr[\text{Count} < \psi \ell] \leq \exp\left(-\frac{\delta^2 \cdot \ell \cdot (1 + p_C)}{4} \right)$$

For sufficiently large ℓ, this probability becomes negligible. Therefore, Algorithm 3 outputs $\gamma = 1$ for matching keys with high probability.

2. Soundness:

For non-matching keys, $\omega_{1,i}$ and $\omega_{2,j}$ are essentially independent random bit strings. Thus, for each position t, $\Pr[\omega_{1,i}[t] = \omega_{2,j}[t]] = \frac{1}{2}$.

The expected value of the count is

$$\mathbb{E}[\text{Count}] = \frac{\ell}{2}$$

Using the Chernoff bound,

$$\Pr[\text{Count} \geq \psi \ell] \leq \exp\left(-2(\psi - \tfrac{1}{2})^2 \ell \right)$$

For $\psi > \frac{1}{2}$, this probability becomes negligible for sufficiently large ℓ. Therefore, Algorithm 3 outputs $\gamma = 0$ for non-matching keys with high probability.

This completes the proof of Theorem 1. □

3.3. Correctness and Security Analysis

Based on the ideal functionality $\mathcal{F}_{\mathsf{PSI}}$ and the semi-honest security definitions introduced in Section 2.2, we now provide the correctness and formal verification that our proposed protocol $\Pi_{\text{Sig-PSI}}$ securely realizes $\mathcal{F}_{\mathsf{PSI}}$.

Theorem 2 (Correctness). *All operations are over $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n+1)$. Let $H : \{0,1\}^* \to \mathcal{R}_q$ be modeled as a random oracle. Secrets and errors are sampled i.i.d. from a discrete Gaussian χ_α. Let the sender hold $X = \{x_i\}_{i\in[n_1]}$, the receiver hold $Y = \{y_j\}_{j\in[n_2]}$, and the intersection $I^\star = X \cap Y$. Based on the properties of OKVS and $Compare(\cdot)$, the receiver's output $I = Compare(\omega_1, \omega_2, \mathbf{K}_2)$ in $\Pi_{Sig\text{-}PSI}$ equals $I^\star$ except with negligible probability in λ.*

Proof of Theorem 2. For any $y_j \in Y$, we have

$$OKVS.Decode(S, h(y_j)) = \mathbf{P}_{1,i}. \tag{10}$$

(1) If $y_j = x_i \in X$,

$$OKVS.Decode(S, h(y_j)) = \mathbf{P}_{1,i} = H(y_j)\mathbf{s}_1 + 2\mathbf{e}_{1,i}. \tag{11}$$

Then,

$$\begin{aligned} \mathbf{K}_{1,j} &= \mathbf{P}_{2,j}\mathbf{s}_1 + 2\mathbf{g}_{1,j} = H(y_j)\mathbf{s}_1\mathbf{s}_2 + 2(\mathbf{e}_{2,j}\mathbf{s}_1 + \mathbf{g}_{1,j}) \bmod q, \\ \mathbf{K}_{2,j} &= \mathbf{P}_{1,i}\mathbf{s}_2 + 2\mathbf{g}_{2,j} = H(y_j)\mathbf{s}_1\mathbf{s}_2 + 2(\mathbf{e}_{1,i}\mathbf{s}_2 + \mathbf{g}_{2,j}) \bmod q, \end{aligned} \tag{12}$$

and therefore

$$\mathbf{K}_{1,j} \;=\; \mathbf{K}_{2,j} + \xi \bmod q, \qquad \xi \;=\; 2\big(\mathbf{e}_{2,j}\mathbf{s}_1 + \mathbf{g}_{1,j} - \mathbf{e}_{1,i}\mathbf{s}_2 - \mathbf{g}_{2,j}\big). \tag{13}$$

Each coefficient of ξ is a sum of the independent discrete Gaussian; standard variance calculations give $\sqrt{\mathrm{Var}(\xi[t])} \le \alpha_\xi = \sqrt{8n\alpha^4 + 8\alpha^4}$. With $h_1 = 5\alpha_\xi$, the tail bound $\Pr[\|\xi\|_\infty > h_1] \le 2^{-21.73}$ per coefficient. If the per-coefficient noise is bounded by h_1 and the robustness of SigMa,

$$\omega_{1,j} = \mathrm{Sig}(\mathbf{K}_{1,j}) = \mathrm{Sig}(\mathbf{K}_{2,j}) = \omega_{2,j}. \tag{14}$$

That is, every intersection element will be accepted by $Compare\big(\omega_{1,j}, \omega_{2,j}\big) = 1$.

(2) If $y_j \notin X$, then $\mathbf{P}'_1(y_j) = OKVS.Decode(S, h(y_j))$ is pseudorandom and independent of $\mathbf{P}_{1,i} (i \in [n_1])$. Then, $\mathbf{K}_{1,j}$ and $\mathbf{K}_{2,j}$ behave (nearly) independently across coefficients, and the chance that $Sig(\cdot)$ outputs identical signals on a non-match is negligible. □

Theorem 3 (Honest but Curious Security against Receiver in the protocol $\Pi_{\text{Sig-PSI}}$). *There exists a PPT simulator SIM_B such that for all security parameters λ and inputs X, Y, letting $I = X \cap Y$ and $n_1 = |X|$,*

$$\mathrm{REAL}^{\lambda,\text{Receiver}}_{\Pi_{Sig\text{-}PSI}}(X, Y) \;\approx\; \mathrm{SIM}_B(1^\lambda,\, Y,\, n_1,\, I), \tag{15}$$

*under the RLWE assumption in $\mathcal{R}_q$, the random-oracle model for H, the unpredictability of **OKVS** on non-programmed keys, and the robustness of* Sig.

Proof of Theorem 3. We describe the simulator algorithm $\mathrm{SIM}_B(1^\lambda, Y, n_1, I)$ in Algorithm 5. We argue that $\mathrm{REAL}^{\lambda,\mathrm{Receiver}}_{\Pi_{\text{Sig-PSI}}}(X, Y) \approx \mathrm{SIM}_B(1^\lambda, Y, n_1, I)$ using a multi-step hybrid argument, where each neighboring pair of hybrid distributions is computationally indistinguishable.

Algorithm 5: The simulator algorithm $\mathrm{SIM}_B(1^\lambda, Y, n_1, I)$.

Input: 1^λ,Y,n_1, I

1 Sends a random OKVS S which encodes n_1 key–value pairs as Sender's message in step 3.
2 Samples $\mathbf{s}_2, \mathbf{e}_{2,j} \leftarrow \chi_\alpha$ for $j \in [n_2]$.
3 For each $j \in [n_2]$, honestly computes $\mathbf{P}_{2,j} = H(y_j)\mathbf{s}_2 + 2\mathbf{e}_{2,j} \bmod q$, and sends $\{\mathbf{P}_{2,j}\}_{j\in I}$ as Receiver's message in step 5.
4 For each $j \in [n_2]$, honestly computes $\mathbf{P}'_1(y_j) = \mathbf{OKVS.Decode}(S, h(y_j))$, $\mathbf{K}_{2,j} = \mathbf{P}'_1(y_j)\mathbf{s}_2 + 2\mathbf{g}_{2,j} \pmod q$ and $\omega_{2,j} = Sig(K_{2,j})$, where $\mathbf{g}_{2,j}$ is randomly sampled from χ_α.
5 Computes $\{j \in [n_2] : y_j \in I\} = J \subseteq [n_2]$. Create $\{\omega_{1,j}\}_{j\in[n_2]}$ as Sender's message in step 6 as follows, for $j \in [n_2] \setminus J$, let $\omega_{1,j} = Sig(\mathsf{h}_{1,j})$ where $\mathsf{h}_{1,j}$ is randomly selected from $\mathcal{R}_q$; for $j \in J$, let $\omega_{1,j} = \omega_{2,j}$.
6 Outputs Receiver's view in the simulated execution above.

Hyb_0 : The view of Receiver in the real execution of the protocol $\Pi_{\text{Sig-PSI}}$.
Hyb_1 : The same as Hyb_0, except in step 3, the OKVS S sent by the Sender is replaced with a random OKVS that encodes n_1 key–value pairs. Note that the values of encoding in the real execution, $\mathbf{P}_{1,i} = H(x_i)\mathbf{s}_1 + 2\mathbf{e}_{1,i} \bmod q$, are RLWE samples with key s_1 held by the Sender. Therefore, if H is modeled as a random oracle, the doubly obliviousness of OKVS and the RLWE assumption in $\mathcal{R}_q$ holds, then Hyb_0 and Hyb_1 are computationally indistinguishable.
Hyb_2 : The same as Hyb_1, except in step 7, all $\mathbf{P}'_1(y_j)$, $\mathbf{K}_{2,j}$ and $\omega_{2,j}$ are computed using S in Hyb_2 instead of S sent by the Sender in the real execution. If the random decoding property of OKVS and the RLWE assumption in $\mathcal{R}_q$ holds, then Hyb_1 and Hyb_2 are computationally indistinguishable.
Hyb_3 : The same as Hyb_2, except in step 6, those $\omega_{1,j}(j \in [n_2] \setminus J)$ sent by the Sender are replaced with $\omega_{1,j} = Sig(\mathbf{h}_{1,j})$ where $\mathbf{h}_{1,j}$ are random elements from $\mathcal{R}_q$. Note that $\mathbf{K}_{1,i} = \mathbf{P}_{2,j}\mathbf{s}_1 + 2\mathbf{g}_{1,i} \bmod q$ is an RLWE sample with key s_1 held by the Sender. Therefore, if the RLWE assumption in $\mathcal{R}_q$ holds, then Hyb_2 and Hyb_3 are computationally indistinguishable.
Hyb_4 : The same as Hyb_3, except in step 6, those $\omega_{1,j}(j \in J)$ sent by the Sender are replaced with $\omega_{1,j} = \omega_{2,j}$. If the robustness of Sig and the RLWE assumption in $\mathcal{R}_q$ holds, then Hyb_3 and Hyb_4 are computationally indistinguishable.

Note that Hyb_4 is the view of the Receiver output by $\mathrm{SIM}_B(1^\lambda, Y, n_1, I)$, and thus the proof completes. □

Theorem 4 (Honest but Curious Security against Sender in the protocol $\Pi_{\text{Sig-PSI}}$)**.** *There exists a PPT simulator SIM_A such that for all security parameters λ and inputs X, Y,*

$$REAL^{\lambda,Sender}_{\Pi_{Sig\text{-}PSI}}(X, Y) \approx SIM_A(1^\lambda, X, n_2), \tag{16}$$

where $n_2 = |Y|$.

Proof of Theorem 4. We describe the simulator algorithm $\mathrm{SIM}_A(1^\lambda, X, n_2)$ in Algorithm 6. We argue that $\mathrm{REAL}^{\lambda,\mathrm{Sender}}_{\Pi_{\text{Sig-PSI}}}(X, Y) \approx \mathrm{SIM}_A(1^\lambda, X, n_2)$ using a multi-step hybrid argument, where each neighboring pair of hybrid distributions is computationally indistinguishable.

Algorithm 6: The simulator algorithm $\mathrm{SIM}_A(1^\lambda, X, n_2)$.

Input: 1^λ, X, n_2

1 Samples $\mathbf{s}_1, \mathbf{e}_{1,i} \leftarrow \chi_\alpha$ for $i \in [n_1]$.
2 Simulate step 2 and 3 honestly.
3 For each $j \in [n_2]$, randomly selects $\mathbf{P}_{2,j}$ from $\mathcal{R}_q$ and sends $\{\mathbf{P}_{2,j}\}_{j\in[n_2]}$ to Sender as Receiver's message in step5.
4 For each $j \in [n_2]$, samples $\mathbf{g}_{1,j} \leftarrow \chi_\alpha$, honestly computes $\mathbf{K}_{1,j} = \mathbf{P}_{2,j}\mathbf{s}_1 + 2\mathbf{g}_{1,j} \bmod q$ and $\omega_{1,j} = Sig(\mathbf{K}_{1,j})$.
5 Sends $\{\omega_{1,j}\}_{j\in[n_2]}$ in the order corresponding to $\{\mathbf{P}_{2,j}\}_{j\in[n_2]}$ as Sender's message in step 6.
6 Outputs Sender's view in the simulated execution above.

Hyb_0 : The view of the Sender in the real execution of the protocol $\Pi_{\text{Sig-PSI}}$.
Hyb_1 : The same as Hyb_0, except in step 5, all $\mathbf{P}_{2,j}$ sent by the Receiver are replaced with random elements $\mathbf{P}_{2,j}$ from $\mathcal{R}_q$. Note that $\mathbf{P}_{2,j} = H(y_j)\mathbf{s}_2 + 2\mathbf{e}_{2,j} \bmod q$ are RLWE samples with key $\mathbf{s}_2$ held by the Receiver. Therefore, if H is modeled as a random oracle and the RLWE assumption in $\mathcal{R}_q$ holds, then Hyb_0 and Hyb_1 are computationally indistinguishable.
Hyb_2 : The same as Hyb_1, except in step 6, all $\mathbf{K}_{1,j}$ and $\omega_{1,j}$ are computed using $\mathbf{P}_{2,j}$ in Hyb_1 instead of $\mathbf{P}_{2,j}$ sent by the Receiver in the real execution, and these newly generated $\omega_{1,j}$ are sent to the Receiver in step 6. If the RLWE assumption in $\mathcal{R}_q$ holds, then Hyb_1 and Hyb_2 are computationally indistinguishable.

Note that Hyb_2 is the view of the Sender output by $\mathrm{SIM}_A(1^\lambda, X, n_2)$, and thus the proof completes. □

4. Optimization on Implementation of OKVS

In Step 3 of our proposed PSI in Figure 2, the sender encodes a set of key–value pairs $(k_i, v_i) := (h(x_i), \mathbf{P}_{1,i})$ to obtain an OKVS, i.e., $S \leftarrow \mathbf{OKVS.Encode}(\{(h(x_i), \mathbf{P}_{1,i})\}_{i\in[n_1]})$, where each $\mathbf{P}_{1,i}$ is a polynomial of degree d over a ring $\mathcal{R} = \mathbb{Z}_q[x]/(x^n+1)$, with d coefficients. For simplicity, we denote the d coefficients of $\mathbf{P}_{1,i}$ as $\mathbf{P}_{1,i} = v_i = (v_{i,1}, \ldots, v_{i,d})$. We observe that directly plugging the most efficient OKVS construction from [26] into our protocol will lead to the following issues. First, it requires each value v_i to belong to a finite field $\mathbb{F}$, whereas our polynomial coefficients lie in $\mathbb{Z}_q^d$. Concatenating the d coefficients into a single value would result in a length of $\log_2 q \cdot d$, exceeding the maximum allowed value length in the OKVS. Second, treating each coefficient independently would cause the same key k_i to be associated with d distinct values, violating the requirement that all keys must be distinct. To ensure all keys remain unique while keeping each value within the allowed length, we introduce a composite key–value pair generation procedure. This method maps one original key–value pair $(k_i, v_i = (v_{i,1}, \ldots, v_{i,d}))$ into d derived key–value pairs $(\hat{k}_{i,j}, v_{i,j})$ for $j \in [d]$, where each key is distinct. Specifically, we define $\hat{k}_{i,j} = \mathrm{Hash}(j \parallel k_i)$. This results in a total of $N = d \cdot n_1$ distinct key–value pairs, which can be efficiently encoded in a single OKVS structure. During the decoding phase, for a query key k, the corresponding derived keys $\hat{k}_j$ for $j \in [d]$ are computed, allowing all $v_{i,j}$ to be retrieved efficiently.

4.1. Generation of Composite Key–Value Pairs

Let $\mathcal{K}$ be the key space and $\mathcal{R}$ a coefficient ring (e.g., $\mathbb{Z}_q$). Each database item is $\mathcal{D}_i = (k_i, \mathbf{v}_i)$, where $k_i \in \mathcal{K}$ and $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,d}) \in \mathcal{R}^d$ are the d coefficients of polynomial i. Our OKVS materializes d scalar pairs per item by defining a composite key

$$\widehat{k}_{i,j} \;=\; H(j, k_i) \;=\; \mathsf{Hash}\big(\mathsf{le32}(j) \,\|\, k_i\big) \quad \text{for } j \in [d],$$

where $\mathsf{le32}(j)$ is a fixed-length little-endian encoding of $j \in [d]$. This ensures that each $\widehat{k}_{i,j}$ is related to j and all $\widehat{k}_{i,1}, \cdots, \widehat{k}_{i,d}$ are distinct. Therefore, each $\mathcal{D}_i = (k_i, \mathbf{v}_i)$ is converted into d distinct pairs $\{(\widehat{k}_{i,j}, v_{i,j})\}_{j\in[d]}$. The procedure is summarized in Algorithm 7.

Algorithm 7: OKVS.CompositeKVGen

Input : Database $\mathcal{D} = \{(k_i, \mathbf{v}_i)\}_{i=1}^{n}$ with $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,d}) \in \mathcal{R}^d$; fixed dimension d; fixed-length little-endian encoder $\mathsf{le32}(\cdot)$; hash function $\mathsf{Hash}(\cdot)$.

Output: $\mathcal{D}' = \{(\widehat{k}_{i,j}, v_{i,j}) \mid i \in [n], j \in [d]\}$.

$\mathcal{D}' \leftarrow \emptyset$ **for** $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** d **do**
    $J_{\text{bytes}} \leftarrow \mathsf{le32}(j)$
    $\widehat{k}_{i,j} \leftarrow \mathsf{Hash}(J_{\text{bytes}} \,\|\, k_i)$ Insert $(\widehat{k}_{i,j}, v_{i,j})$ into $\mathcal{D}'$
  end
end
return $\mathcal{D}'$

4.2. Encoding of OKVS

Given database items $\mathcal{D} = \{(k_i, \mathbf{v}_i)\}_{i=1}^{n}$ with fixed polynomial dimension d (e.g., $d = 128$), encoding proceeds by first invoking *OKVS.CompositeKVGen* (Algorithm 7) to materialize the scalarized set $\mathcal{D}' = \{(\widehat{k}_{i,j}, v_{i,j})\}$ under composite keys $\widehat{k}_{i,j} = \mathsf{Hash}(\mathsf{le32}(j) \,\|\, k_i)$. Let $N = |\mathcal{D}'| = dn$. We then set OKVS parameters with redundancy $\epsilon > 0$, choose a bandwidth w for security parameter λ), fix $m = (1+\epsilon)N$, and sample public randomness $R = (R_1, R_2)$. Finally, our *OKVS.Encode* (Algorithm 8) directly calls OKVS encoder $\mathsf{Enc}_{\mathrm{RB}}$ (Algorithm 1) to solve the banded linear system and produce the encoding array $S \in \mathbb{F}^m$.

Algorithm 8: OKVS.Encode (using OKVS encoding algorithm $\mathsf{Enc}_{\mathrm{RB}}$ of Algorithm 1).

Input : $\mathcal{D} = \{(k_i, \mathbf{v}_i)\}_{i=1}^{n}$, dimension d; OKVS params: redundancy $\epsilon > 0$, security λ, band width w; field $\mathbb{F}$; OKVS randomness $R = (R_1, R_2)$.

Output: OKVS encoding $S \in \mathbb{F}^m$.

$\mathcal{D}' \leftarrow \mathsf{OKVS.CompositeKVGen}(\mathcal{D}, d)$ // Algorithm 7
$N \leftarrow |\mathcal{D}'|$ // $N = dn$
$m \leftarrow (1+\epsilon)\, N$
$S \leftarrow \mathsf{Enc}_{\mathrm{RB}}(\mathcal{D}', m, w, R)$ // Algorithm 1
return S

4.3. Decoding of OKVS

Given an encoding array S and a queried key k, we reconstruct the full coefficient vector by first obtaining the composite keys $\{\widehat{k}_j\}_{j\in[d]]}$ from k. Our *OKVS.Decode* (Algorithm 9) then invokes the OKVS decode algorithm $\mathsf{Dec}_{\mathrm{RB}}$ (Algorithm 2) independently for each $\widehat{k}_j$ and assembles $\mathbf{v} = (v_1, \ldots, v_d)$.

Algorithm 9: OKVS.Decode (using OKVS decoding algorithm $\mathsf{Dec}_{\mathrm{RB}}$ of Algorithm 2)

Input : OKVS structure S; query key $k \in \mathcal{K}$; dimension d.
Output: Reconstructed vector $\mathbf{v} = (v_1, \ldots, v_d) \in \mathcal{R}^d$.

Initialize $\mathbf{v} \leftarrow (\perp, \ldots, \perp)$
for $j \leftarrow 1$ **to** d **do**
  $J_{\mathrm{bytes}} \leftarrow \mathsf{le32}(j)$
  $\widehat{k}_j \leftarrow \mathsf{Hash}(J_{\mathrm{bytes}} \,\|\, k)$
  $v_j \leftarrow \mathsf{Dec}_{\mathrm{RB}}(S, \widehat{k}_j)$ `// Algorithm 2`
end
return $\mathbf{v}$

4.4. Correctness

The correctness of our OKVS scheme is inherited from the correctness of underlying OKVS construction [26]. For parameters $m = (1+\epsilon)N$ and $w = O(\lambda/\epsilon + \log N)$, the random band matrix is solvable with probability at least $1 - 2^{-\lambda}$, ensuring the scalar decoder fails for any individual coefficient with a negligible probability $P_{\mathrm{RB}}(\lambda) \leq 2^{-\lambda}$. In our composite key generation, defined by $\widehat{k}_{i,j} = H(j, k_i)$ for a cryptographic hash function, guarantees that decoding events for different coefficients are independent. Therefore, by the union bound, the probability of failing to decode any coefficient for a given polynomial key k_i is upper-bounded by $d \cdot P_{\mathrm{RB}}$. For a constant dimension d, this probability remains negligible in the security parameter λ_{OKVS}, thereby upholding the overall correctness guarantee. Any decoding failure is confined to a single coefficient and does not affect the recovery of others.

4.5. Complexity

The computational and spatial complexity of our OKVS scheme scales efficiently with the total number of scalar elements $N = d \cdot n$. The composite key generation is performed in linear time, $O(N)$. The cost of encoding is dominated by the $O(Nw) = O(dnw)$ operations required for the band-aware Gaussian elimination algorithm of the underlying OKVS. Decoding on a queried key is a constant-time operation relative to the bandwidth, requiring $O(w)$ time. Consequently, reconstructing a full d-dimensional polynomial sequentially requires $O(dw)$ time. A key practical advantage is the potential for parallelization. Since decoding each coefficient is independent, the wall-clock time for full polynomial reconstruction can be reduced to nearly $O(w)$ in a multi-core environment. The storage overhead is determined by the encoding length $m_{\mathsf{poly}} = (1+\epsilon)dn$, indicating linear growth in the size of the OKVS structure with respect to N.

5. Performance

5.1. Implementation

We implement our PSI protocol in C++, following the design in Figure 2. Private elements are hashed using SHA-256, and the OKVS Encode/Decode routines are adapted from the open-source implementation (https://github.com/google/private-membership/tree/main/research/okvs (accessed on 14 December 2025)).

All experiments are conducted with the statistical security parameter $\lambda = 128$ and the OKVS-specific parameter $\lambda_{\mathrm{OKVS}} = 40$, the latter controlling the decoding failure probability of the OKVS construction.

The experiments are executed on a server with dual 2.0 GHz Intel Xeon Gold 8470 CPUs and 1024 GB RAM. Network conditions are emulated as a 10 Gbps LAN with 0.2 ms round-trip latency using Linux `tc`. For comparison, we evaluate post-quantum PSI proto-

cols based on FHE [22] and lattice-based OPRF [23] in the same environment, using their publicly available implementations (https://github.com/Microsoft/APSI (accessed on 14 December 2025), https://github.com/meyira/leap (accessed on 14 December 2025)).

All implementations are tested under both single-threaded and multi-threaded settings with $T \in \{1, 32\}$, and reported values are averaged over 10 runs. In the following tables, the best-performing protocols in terms of runtime and communication are highlighted.

Our protocol supports an offline–online execution paradigm:

- Offline phase: The sender hashes its input set, samples randomness, and computes public keys and encoded into an OKVS encoded structure. Since costs scale with the sender's set size, offline time is measured as the sender's computation time.
- Online phase: Upon receiving the receiver's input, both parties compute signals in parallel, and the receiver determines the intersection using a signal matching mechanism. Online time is defined as the receiver's total computation time, including hashing, key generation, OKVS decode, signal computation, and signal matching.

The total runtime equals the sum of the sender's offline time and the receiver's total time. The communication overhead is partitioned accordingly: the OKVS encoded structure sent by the sender in the offline phase, and the receiver's public keys plus sender's signals in the online phase.

We evaluate protocol performance under varying set sizes. For symmetric size inputs, we consider $n_1 = n_2 \in \{2^8, 2^{12}, 2^{16}, 2^{20}\}$. For asymmetric size inputs, we fix $n_1 \in \{2^{12}, 2^{16}, 2^{20}, 2^{24}\}$ and $n_2 \in \{2^7, 2^8, 2^{10}, 2^{12}, 2^{16}, 2^{20}\}$, executing all combinations with $n_2 \leq n_1$.

To instantiate OKVS for our PSI protocol, we require concrete band length parameters (n_1, ϵ) under $\lambda_{\text{OKVS}} = 40$. Table 1 reports the values used in our experiments. Since the band length parameters given in [26] cannot be used directly, the data in Table 1 are calculated by linear regression of the published data points, following the method outlined in Section 6.1 of [26] (details are provided in Appendix A).

The parameter ϵ controls the slack in the OKVS band length: a larger ϵ yields a longer band and thus higher communication but lower encoding/decoding cost, whereas a smaller ϵ shortens the band and reduces communication at the price of increased computation.

We employ Number Theoretic Transforms (NTTs) to expedite polynomial multiplication within DingKE. The key exchange parameters are set as follows: $n = 128$, $\sigma = 4.19$, $q = 2255041$, and we set $\psi = 0.9$ in Algorithm 3 (detailed theoretical values of ψ are given in Appendix B).

Intuitively, ψ acts as a signal-consistency threshold in our RLWE-based comparison: a larger ψ makes the matching criterion stricter, improving completeness but increasing sensitivity to noise, while a smaller ψ relaxes the criterion, enhancing robustness to random errors but potentially reducing correctness. The choice $\psi = 0.9$ provides a practical balance between these effects in our parameter regime.

Table 1. OKVS band length data for different ϵ and n_1 values.

ϵ \ n_1	2^7	2^8	2^{10}	2^{12}	2^{16}	2^{20}	2^{24}
0.03	555	555	571	593	663	670	722
0.07	250	250	258	268	283	307	331
0.10	184	184	189	197	218	220	231

5.2. Benchmarks

Table 2 summarizes the offline/online running time (single-threaded and multi-threaded) and communication overhead, and performs systematic measurements under different set sizes (n_1, n_2), parallelism $T \in \{1, 32\}$ and OKVS parameters $\varepsilon = 0.10$.

A first observation is that, for a fixed n_2, the online running time is essentially insensitive to n_1. As shown in Table 2, within each fixed n_2 the online time remains nearly flat as n_1 increases from small to very large. Because the online phase is only related to the receiver's data (calculating the signal, public key, and signal matching), its cost depends on n_2 rather than n_1. For example, when $n_2 = 2^7$ and n_1 grows from 2^7 to 2^{20} and $T = 1$, the online time stays within 0.0002 s $\pm$ 0.0002 s. In practice, this implies that even when the sender's dataset varies substantially over time, the online latency remains both efficient and predictable. Using multi-thread can significantly reduce the communication time, particularly in the offline phase. For larger n_1, we observe near-linear speedup with increasing T. As for online time, in large-scale settings (e.g., $n_1 = 2^{24}$, $n_2 = 2^{20}$), when T increases from 1 to 32, the online time decreases by 1.36–1.42×, from 12.98–13.92 s to 9.57–9.78 s, while the total time decreases by 1.29–1.38×.

Table 2. Runtime and communication overhead for all phases of our improved OKVS-based PSI protocol. T represents the number of threads used by the sender and receiver simultaneously.

n_1	n_2	Sender Offline (s)		Online (s)		Total (s)		Comm. (MB)	
		T		T		T			
		1	32	1	32	1	32	Offline	Online
2^{12}	2^7	0.039	0.025	0.001	0.001	0.040	0.026	1.65	0.06
	2^8	0.040	0.026	0.003	0.002	0.043	0.028	1.65	0.13
	2^{10}	0.040	0.026	0.011	0.005	0.051	0.031	1.65	0.52
	2^{12}	0.040	0.026	0.044	0.025	0.084	0.051	1.65	2.06
2^{16}	2^8	0.653	0.433	0.004	0.003	0.657	0.436	26.40	0.13
	2^{10}	0.654	0.434	0.013	0.008	0.667	0.442	26.40	0.52
	2^{12}	0.657	0.431	0.047	0.025	0.703	0.455	26.40	2.06
	2^{16}	0.653	0.432	0.781	0.428	1.433	0.860	26.40	33
2^{20}	2^8	10.162	7.437	0.005	0.004	10.167	7.441	422.40	0.13
	2^{12}	10.190	7.227	0.063	0.048	10.253	7.275	422.40	2.06
	2^{16}	10.141	7.298	0.797	0.458	10.938	7.756	422.40	33
	2^{20}	10.142	6.994	12.455	7.632	22.597	14.626	422.40	528
2^{24}	2^8	177.559	129.876	0.005	0.005	177.564	129.881	4096	0.13
	2^{12}	177.785	127.669	0.075	0.083	177.860	127.752	4096	2.06
	2^{16}	176.905	136.517	1.081	0.945	177.986	137.462	4096	33
	2^{20}	177.374	128.702	12.976	9.566	190.350	138.268	4096	528

The same decoupling shows up in the online communication. When n_2 is fixed, the online communication overhead does not change with n_1. Because only the sender's signals and receiver's public key are transmitted, both are proportional to n_2. As shown in Table 2, regardless of the value of n_1, the online communication overhead is always 2.06 MB when $n_2 = 2^{12}$, and is fixed at 33 MB and 528 MB for $n_2 = 2^{16}$ and 2^{20}, respectively. Furthermore, the offline communication cost grows roughly linearly with n_1: offline communication rises from about 1–2 MB when $n_1 \leq 2^{12}$ to roughly 24–26 MB, and near 4 GB at 2^{24}.

Overall, Table 2 shows a consistent picture: the online phase scales with n_2 and is essentially independent of n_1 in both time and communication; the offline phase scales with n_1 and benefits strongly from parallelism.

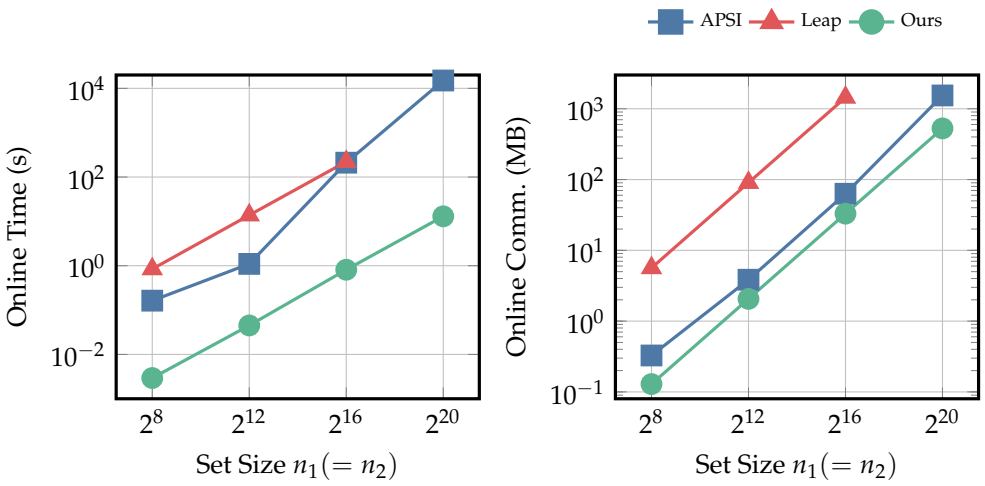
5.3. Comparison with APSI [22] and Leap [23] in Symmetric Set Sizes

We compare our protocol with APSI [22] and Leap [23] under symmetric set sizes, i.e., $n_1 = n_2 \in \{2^8, 2^{12}, 2^{16}, 2^{20}\}$, as shown in Table 3. Unless stated otherwise, we report single-thread results ($T = 1$) for fairness: Leap's open-source implementation is single-threaded (columns for $T = 32$ are unavailable), and it fails with out-of-memory at 2^{20} as reflected by missing entries.

Table 3. OKVS-based performance comparison of protocols for symmetric set sizes. T represents the number of threads used by the sender and receiver simultaneously. The purple highlights indicate the best performance for each parameter setting among the different protocols.

		Sender Offline (ms)		Online (ms)		Total Time (ms)		Comm. (MB)	
$n_1 = n_2$	T	1	32	1	32	1	32	Offline	Online
2^8	APSI [22]	17	30	163	32	219	99	0	0.327
	Leap [23]	429	-	842	-	1271	-	10.220	5.642
	Ours	2.966	3.862	2.934	1.288	5.9	5.15	0.104	0.129
2^{12}	APSI [22]	222	248	1100	218	1670	816	0	3.843
	Leap [23]	6630	-	13,803	-	20,433	-	152.736	90.281
	Ours	40.317	26.559	45.003	22.119	85.32	48.678	1.651	2.062
2^{16}	APSI [22]	3192	3190	210,581	8628	219,225	17,268	0	63.024
	Leap [23]	107,806	-	221,834	-	329,640	-	2432.999	1444.500
	Ours	655.931	426.576	812.003	401.003	1467.934	827.579	26.401	33.000
2^{20}	APSI [22]	122,949	120,476	15,028,196	606,872	15,255,609	1,013,603	0	1542.29
	Leap [23]	1,712,179	-	-	-	-	-	38,917.204	-
	Ours	10,179.4	7011.05	12,992.003	6397.003	23,171.403	13,408.053	422.401	528.000

As illustrated in Figure 4a, across all tested sizes, our protocol is consistently faster than both APSI [22] and Leap. In small set size, i.e., $n_1 = n_2 = 2^8$, our 2.934 ms is 55.56× faster than APSI's 163 ms and 286.98× faster than Leap's 842 ms. At 2^{16}, our 812.003 ms improves over APSI (210,581 ms) and Leap (221,834 ms) by 259.34× and 273.19×, respectively. At 2^{20}, we finish in 12,992.003 ms, a 1156.73× reduction relative to APSI 15,028,196 ms. Likewise, the total time results show that our protocol is optimal in all tests. Across symmetric sizes 2^8–2^{20}, our total time is faster by 19.57×–658.38× than APSI and by 215.42×–239.49× than Leap.



(**a**) Online time (T = 1) (**b**) Online communication

Figure 4. Performance comparison for symmetric set sizes ($n_1 = n_2$) with single-thread execution ($T = 1$). APSI and Leap correspond to Table 3 baselines.

As illustrated in Figure 4b, our online communication is consistently the lowest across symmetric sizes, achieving roughly 2–3× lower bandwidth than APSI and 44× lower than

Leap wherever Leap reports results. At 2^{20}, Leap is unavailable, and we still transmit 2.92× less than APSI.

5.4. Comparison with APSI [22] & Leap [23] in Asymmetric Set Sizes

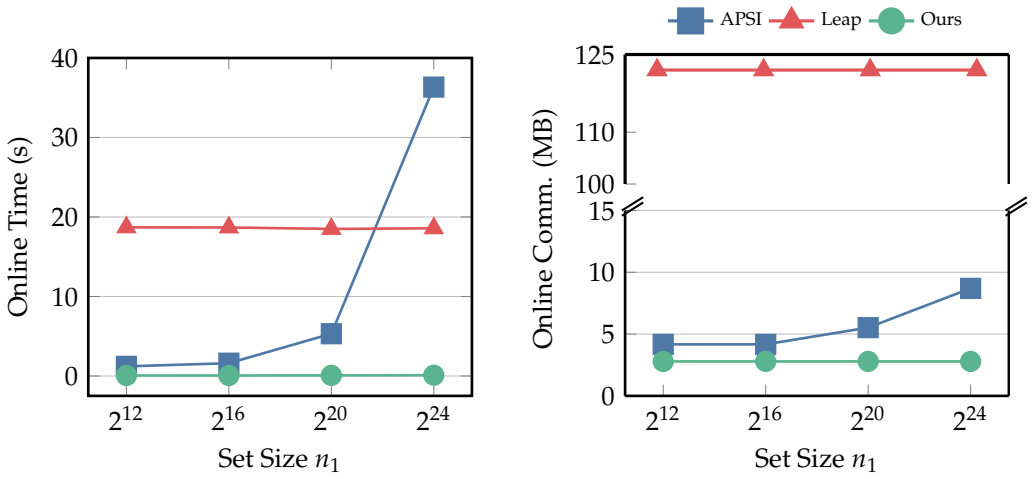
Our PSI protocol delivers the lowest online running time among all compared schemes, and its online cost essentially depends only on the size of the receiver's set n_2. As shown in Figure 5a of the asymmetric setting, regardless of whether n_1 is small or large, our online time remains in the range 0.063–0.116 s for n_2=5535. The large-scale case $n_1 = 2^{24}$ further highlights this gap: at $n_2 = 5535$, our protocol is about 360× and 184× faster than the FHE-based [22] and post-quantum OPRF-based [23] PSI protocols, respectively. The results in Table 4 confirm the same trend against APSI and Leap: for example, at $(2^{12}, 11041)$, our scheme achieves an online time of 0.132 s, which is 19.2× and 282.3× faster than APSI (2.532 s) and Leap (37.265 s); even at $(2^{20}, 11041)$ the online time is still only 0.150 s, yielding 45.5× and 246.4× improvements over APSI and Leap. Overall, these results show that the online phase of our protocol mainly scales with n_2 and remains around 0.1 s across a wide range of parameters.

Table 4. OKVS-based improved performance comparison of protocols for asymmetric set sizes under single-thread execution. The purple highlights indicate the best performance for each parameter setting among the different protocols.

n_1	n_2	Protocol	Sender Offline (s)	Online (s)	Total Times (s)	Offline Comm. (MB)	Online Comm. (MB)
2^{12}	11041	APSI [22]	0.279	2.532	2.811	0	7.805
		Leap [23]	17.940	37.265	55.205	410.463	243.358
		Ours	0.038	0.132	0.170	1.651	5.560
	5535	APSI [22]	0.195	1.213	1.408	0	4.167
		Leap [23]	9.033	18.698	27.731	206.137	121.999
		Ours	0.039	0.068	0.107	1.651	2.787
2^{16}	11041	APSI [22]	1.890	2.942	4.832	0	7.806
		Leap [23]	18.388	37.287	55.675	410.708	243.358
		Ours	0.657	0.127	0.783	26.40	5.560
	5535	APSI [22]	1.788	1.611	3.399	0	4.169
		Leap [23]	9.452	18.682	28.134	206.383	121.999
		Ours	0.654	0.063	0.717	26.40	2.787
2^{20}	11041	APSI [22]	28.229	6.828	35.057	0	9.162
		Leap [23]	24.556	36.956	61.512	414.635	243.358
		Ours	10.181	0.150	10.331	422.40	5.560
	5535	APSI [22]	27.969	5.319	33.288	0	5.522
		Leap [23]	15.729	18.497	34.226	210.309	121.998
		Ours	10.168	0.083	10.251	422.40	2.787
2^{24}	11041	APSI [22]	746.775	45.438	792.213	0	14.925
		Leap [23]	127.165	36.943	164.108	477.461	243.358
		Ours	176.677	0.201	176.878	4096	5.560
	5535	APSI [22]	775.462	36.336	811.798	0	8.696
		Leap [23]	118.303	18.579	136.882	273.135	121.998
		Ours	176.265	0.101	176.366	4096	2.787

To further understand where this small latency comes from, we analyze the receiver's online phase for the largest asymmetric parameters (e.g., $(n_1, n_2) = (2^{24}, 5535)$). Specifically, approximately 50% of the online time is spent on OKVS decoding, about 45% on hashing and NTT-based public-key generation and signal computation, and less than 5% on the final signal-matching step that identifies the intersection. This allocation remains stable across different configurations, and similar proportions are observed for smaller set sizes.

In terms of communication, our protocol incurs the lowest and constant online communication with respect to n_1, depending only on n_2. As shown in Figure 5b of the asymmetric setting, our online bandwidth is a constant 2.787 MB for $n_2 = 5535$, whereas the FHE-based PSI requires 4.167–8.696 MB and the post-quantum OPRF-based PSI uses 121.999 MB. Table 4 shows a similar advantage over APSI and Leap: for $n_2 = 11041$, our protocol needs 5.560 MB online, compared with APSI's 7.805 MB and Leap's 243.358 MB; when $n_2 = 5535$, the communication further reduces to 2.787 MB, outperforming APSI by about 33× and Leap by more than 40×. Taken together, these results confirm that our PSI achieves the *lowest* online time and online communication overhead among all considered baselines.



(**a**) Online time (**b**) Online communication

Figure 5. Online performance comparison at $n_1 = \{2^{12}, 2^{16}, 2^{20}, 2^{24}\}$ and $n_2 = 5535$ with single-thread execution ($T = 1$). APSI and Leap correspond to Table 4 baselines.

It should be noted, however, that this online performance gain is accompanied by higher offline communication costs. While the offline communication remains modest at smaller scales (e.g., around 1.6 MB for $(2^{12}, 11041)$ and about 25 MB for $(2^{16}, 11041)$), it grows significantly at large set sizes, reaching 395–422 MB for $(2^{20}, 11041)$ and approximately 4 GB at $(2^{24}, 11041)$. This indicates that, although our method attains state-of-the-art online performance and communication efficiency, future work should focus on mitigating the offline overhead to further improve scalability.

6. Conclusions

This paper demonstrates that the proposed RLWE-based post-quantum PSI protocol achieves ultra online performance among state-of-the-art post-quantum PSI protocols. The experiments indicate that our proposed PSI's online time is consistently sub-second and nearly invariant to n_1 and ϵ (e.g., $\leq$0.10 s for $n_2 = 5535$ and around 0.2 s for $n_2 = 11041$), while online communication is constant across settings (5.560 MB for $n_2 = 11041$ and 2.787 MB for $n_2 = 5535$). This achieves significant improvements in both computation and communication compared to APSI and Leap in both asymmetric and symmetric schemes.

While our proposed PSI protocol exhibits excellent online performance, it incurs higher offline computation and communication overhead for very large set sizes, which may put pressure on memory and computation time in bandwidth-constrained settings. Thus, we will further consider how to balance the efficiency of the online and offline phases in the future. In parallel, we plan to extend security from semi-honest to malicious. These directions aim to enhance the practicality of post-quantum PSI in production environments and accelerate its adoption under more stringent threat models.

While our proposed PSI protocol exhibits excellent online performance, it incurs higher offline computation and communication overhead for very large set sizes, which may put pressure on memory and running time in bandwidth-constrained settings. We identify this as an inherent trade-off between pre-computation and online efficiency. Thus, our further work will aim to balance the efficiency of the online and offline phases for very large set size. In addition, we plan to extend security from semi-honest to malicious. These directions aim to enhance the practicality of post-quantum PSI in production environments and accelerate its adoption under more stringent threat models.

Author Contributions: Conceptualization, Y.Q. and B.L.; methodology, Y.Q., B.L. and J.D.; software, Y.Q.; validation, Y.Q. and B.L.; formal analysis, Y.Q., H.C. and B.L.; investigation, Y.Q. and B.L.; resources, B.L. and J.D.; data curation, Y.Q.; writing—original draft preparation, Y.Q., H.C. and B.L.; writing—review and editing, Y.Q. and B.L.; visualization, Y.Q. and B.L.; supervision, Y.Q., B.L. and J.D.; project administration, Y.Q., B.L. and J.D.; funding acquisition, J.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key R&D Program of China (No. 2023YFC3305501).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Band Length Calculation Algorithm

The original work of Bienstock et al. [26] provides regression formulas for OKVS parameters, expressed as $\lambda = aw + b$, where (a, b) are best-fit coefficients reported for specific (n, ϵ) configurations, λ is the computational security parameter, and w denotes the band length. However, only a subset of parameter values were explicitly tabulated in their appendix.

Algorithm A1: OKVS Band Length Calculator

Input : $\epsilon \in \{0.03, 0.05, 0.07, 0.10\}$, $n > 0$, $\lambda = 40$
Output: Band length w

Load pre-computed coefficients from *best_fit_lines* dictionary
$ns \leftarrow$ sorted keys of $best_fit_lines[\epsilon]$, $max_experimental_n \leftarrow \max(ns)$
if $n \leq max_experimental_n$ **then**
    foreach *candidate in available_ns* **do**
        if $candidate \geq n$ **then**
            $fit_line \leftarrow best_fit_lines[\epsilon][candidate]$, $a \leftarrow fit_line.a$, $b \leftarrow fit_line.b$
            break
else
    $ns \leftarrow available_ns$, $n_1 \leftarrow ns[-2]$, $n_2 \leftarrow ns[-1]$// Last two data points
    $a_1 \leftarrow best_fit_lines[\epsilon][n_1].a$, $b_1 \leftarrow best_fit_lines[\epsilon][n_1].b$
    $a_2 \leftarrow best_fit_lines[\epsilon][n_2].a$, $b_2 \leftarrow best_fit_lines[\epsilon][n_2].b$
    $\Delta_{\log} \leftarrow \log_2 n_2 - \log_2 n_1$, $a_{slope} \leftarrow (a_2 - a_1)/\Delta_{\log}$, $b_{slope} \leftarrow (b_2 - b_1)/\Delta_{\log}$
    $\Delta_{ext} \leftarrow \log_2 n - \log_2 n_2$, $a \leftarrow a_2 + a_{slope} \times \Delta_{ext}$, $b \leftarrow b_2 + b_{slope} \times \Delta_{ext}$
$w \leftarrow \lceil (\lambda - b)/a \rceil$
return w

For completeness of our evaluation, we reproduce their methodology by extracting the regression coefficients (a, b) and computing the missing band length parameters w for $\lambda_{\text{OKVS}} = 40$. This yields an extended parameter set covering all (n_1, ϵ) required by our experiments. To ensure the reproducibility of the parameter selection process, we provide a complete algorithm for calculating the RB-OKVS band length w as shown in Algorithm A1.

Our implementation is built on a data-driven approach. The coefficients a and b are directly obtained from the experimental best-fit lines shown in Figure 17 of [26]. It also boasts robust extrapolation functionality. When dealing with values of $n = dn_1$ that fall outside the experimentally verified range, we employ linear extrapolation in the logarithmic domain. This allows us to generate reliable estimates for large-scale deployments. Additionally, it can support all relevant ϵ values and automatically handle the entire range of n_1 values, from 2^7 to 2^{24}.

Appendix B. Theoretical Value of ψ

The probability p_C that a random coefficient u falls in $C^Z_{\tau,\eta}$ is

$$p_C = \frac{|\{u \in \mathcal{D}_q : |u| \leq \tau - \eta\}| + |\{u \in \mathcal{D}_q : |u| \geq \tau + \eta\}|}{q} \tag{A1}$$

For large q, this expression can be approximated as $p_C \approx 1 - \frac{4\eta}{q}$. To guarantee both completeness and soundness in Theorem 1, the threshold ψ must satisfy $\frac{1}{2} < \psi < \frac{1+p_C}{2}$.

Using the concrete parameters $\alpha = 4.19$, $n = 128$, $q = 2255041$, we compute the relevant values as follows. First, we obtain $\alpha_{\tilde{\xi}} = \sqrt{8n\alpha^4 + 8\alpha^4} \approx 564.014$, from which $h_1 = 5\alpha_{\tilde{\xi}} \approx 2820.07$ and thus $\eta = \lfloor h_1 \rceil = 2820$. The flip point is $\tau = \lfloor q/4 \rfloor = 563{,}760$, yielding $p_C \approx 0.99500$ and consequently $\frac{1+p_C}{2} \approx 0.9975$. Therefore, the theoretical range for ψ is $0.5 < \psi < 0.9975$.

For practical implementation, we recommend setting $\psi = 0.9$, which provides a good balance between completeness and soundness while allowing for a small margin of error.

References

1. Morales, D.; Franquesa, A.; Domingo-Ferrer, J. Private set intersection: A systematic literature review. *Comput. Sci. Rev.* **2023**, *49*, 100567. [CrossRef]
2. Chen, H.; Laine, K.; Rindal, P. Fast private set intersection from homomorphic encryption. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1243–1255.
3. Falk, B.H.; Noble, D.; Pinkas, B.; Rosulek, M.; Roy, A.; Trieu, N. Private Set Intersection with Linear Communication from General Assumptions. In Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, London, UK, 11 November 2019; pp. 14–25.
4. Kales, D.; Rechberger, C.; Schneider, T.; Senker, M.; Weinert, C. Mobile Private Contact Discovery at Scale. In Proceedings of the 28th USENIX Security Symposium (USENIX Security 19), Santa Clara, CA, USA, 14–16 August 2019; pp. 1447–1464.
5. Demmler, D.; Gusev, L.; Müller-Quade, J.; Schneider, T. PIR-PSI: Scaling Private Contact Discovery. *Proc. Priv. Enhancing Technol.* **2018**, *4*, 159–178. [CrossRef]
6. Ion, M.; Kreuter, B.; Nergiz, A.E.; Patel, S.; Raykova, M.; Saxena, S.; Seth, K.; Shanahan, D.; Yung, M. Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions. IACR ePrint 2017/738. 2017. Available online: https://eprint.iacr.org/2017/738 (accessed on 14 December 2025).
7. Blog, G.S. Helping Organizations Do More Without Collecting More Data (Private Join and Compute). Web Article. 2019. Available online: https://security.googleblog.com/2019/06/helping-organizations-do-more-without-collecting-more-data.html (accessed on 14 December 2025).
8. Trieu, N.; Shehata, K.; Saxena, P.; Shokri, R.; Song, D. Epione: Lightweight contact tracing with strong privacy. *arXiv* **2020**, arXiv:2004.13293. [CrossRef]
9. Chan, J.; Foster, D.; Gollakota, S.; Horvitz, E.; Jaeger, J.; Kakade, S.; Kohno, T.; Langford, J.; Larson, J.; Sharma, P.; et al. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing. *arXiv* **2020**, arXiv:2004.03544. [CrossRef]

10. Pinkas, B.; Schneider, T.; Zohner, M. Faster Private Set Intersection Based on OT Extension. In Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14), San Diego, CA, USA, 20–22 August 2014; pp. 797–812.
11. Kolesnikov, V.; Kumaresan, R.; Rosulek, M.; Trieu, N. Efficient batched oblivious PRF with applications to private set intersection. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 818–829.
12. Pinkas, B.; Schneider, T.; Zohner, M. Scalable private set intersection based on OT extension. *ACM Trans. Priv. Secur. (TOPS)* **2018**, *21*, 1–35. [CrossRef]
13. Pinkas, B.; Rosulek, M.; Trieu, N.; Yanai, A. SpOT-light: Lightweight private set intersection from sparse OT extension. In *Advances in Cryptology–CRYPTO 2019: Proceedings of the 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019;* Proceedings, Part III 39; Springer: Berlin/Heidelberg, Germany, 2019; pp. 401–431.
14. Rosulek, M.; Trieu, N. Compact and malicious private set intersection for small sets. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual, 15–19 November 2021; pp. 1166–1181.
15. Raghuraman, S.; Rindal, P. Blazing Fast PSI from Improved OKVS and Subfield VOLE. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022; CCS '22; pp. 2505–2517.
16. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134.
17. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]
18. Debnath, S.K.; Stănică, P.; Choudhury, T.; Kundu, N. Post-quantum protocol for computing set intersection cardinality with linear complexity. *IET Inf. Secur.* **2020**, *14*, 661–669. [CrossRef]
19. Debnath, S.K.; Choudhury, T.; Kundu, N.; Dey, K. Post-quantum secure multi-party private set-intersection in star network topology. *J. Inf. Secur. Appl.* **2021**, *58*, 102731. [CrossRef]
20. Abudaqa, A.A.; Alshehri, K.; Felemban, M. On the Homomorphic Properties of Kyber and McEliece with Application to Post-Quantum Private Set Intersection. *Cryptography* **2025**, *9*, 66. [CrossRef]
21. Chen, H.; Huang, Z.; Laine, K.; Rindal, P. Labeled PSI from Fully Homomorphic Encryption with Malicious Security. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; CCS '18; pp. 1223–1237.
22. Cong, K.; Moreno, R.C.; da Gama, M.B.; Dai, W.; Iliashenko, I.; Laine, K.; Rosenberg, M. Labeled PSI from Homomorphic Encryption with Reduced Computation and Communication. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual, 15–19 November 2021; CCS '21; pp. 1135–1150.
23. Heimberger, L.; Kales, D.; Lolato, R.; Mir, O.; Ramacher, S.; Rechberger, C. Leap: A Fast, Lattice-based OPRF with Application to Private Set Intersection. In *Advances in Cryptology—EUROCRYPT 2025: 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, 4–8 May 2025*; Proceedings, Part VII; Springer: Cham, Switzerland, 2025; pp. 254–283.
24. Kiss, Á.; Liu, J.; Schneider, T.; Asokan, N.; Pinkas, B. Private Set Intersection for Unequal Set Sizes with Mobile Applications. *Priv. Enhancing Technol.* **2017**, *2017*, 177–197. [CrossRef]
25. Heimberger, L.; Hennerbichler, T.; Meisingseth, F.; Ramacher, S.; Rechberger, C. OPRFs from Isogenies: Designs and Analysis. In Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, Singapore, 1–5 July 2024; ASIA CCS '24; pp. 575–588. [CrossRef]
26. Bienstock, A.; Patel, S.; Seo, J.Y.; Yeo, K. Near-Optimal Oblivious Key-Value Stores for Efficient PSI,PSU and Volume-HidingMulti-Maps. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 9–11 August 2023; pp. 301–318.
27. Meadows, C. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 7–9 April 1986; p. 134.
28. Huberman, B.A.; Franklin, M.; Hogg, T. Enhancing privacy and trust in electronic communities. In Proceedings of the 1st ACM Conference on Electronic Commerce, Denver, CO, USA, 3–5 November 1999; pp. 78–86.
29. Freedman, M.J.; Nissim, K.; Pinkas, B. Efficient private matching and set intersection. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; pp. 1–19.
30. Kissner, L.; Song, D. Privacy-preserving set operations. In *Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 241–257.
31. De Cristofaro, E.; Tsudik, G. Practical Private Set Intersection Protocols with Linear Complexity. In *Financial Cryptography and Data Security*; Sion, R., Ed.; Springer: Berlin/Heidelberg, Germany , 2010; pp. 143–159.
32. Pinkas, B.; Schneider, T.; Segev, G.; Zohner, M. Phasing: Private set intersection using permutation-based hashing. In Proceedings of the 24th USENIX Security Symposium (USENIX Security 15), Washington, DC, USA, 12–14 August 2015; pp. 515–530.

33. Rindal, P.; Rosulek, M. Malicious-secure private set intersection via dual execution. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1229–1242.
34. Chase, M.; Miao, P. Private Set Intersection in the Internet Setting from Lightweight Oblivious PRF. In *Advances in Cryptology—CRYPTO 2020*; Micciancio, D., Ristenpart, T., Eds.; Springer: Cham, Switzerland, 2020; pp. 34–63.
35. Huang, Y.; Evans, D.; Katz, J. Private set intersection: Are garbled circuits better than custom protocols? In Proceedings of the NDSS, San Diego, CA, USA, 5–8 February 2012.
36. Pinkas, B.; Schneider, T.; Weinert, C.; Wieder, U. Efficient circuit-based PSI via cuckoo hashing. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, 26–30 May 2024; Springer: Berlin/Heidelberg, Germany, 2024; pp. 125–157.
37. Pinkas, B.; Schneider, T.; Tkachenko, O.; Yanai, A. Efficient Circuit-Based PSI with Linear Communication. In *Advances in Cryptology—EUROCRYPT 2019*; Ishai, Y., Rijmen, V., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 122–153.
38. Chandran, N.; Gupta, D.; Shah, A. Circuit-PSI with Linear Complexity via Relaxed Batch OPPRF. *Proc. Priv. Enhancing Technol.* **2021**, *2022*, 353–372. [CrossRef]
39. Resende, A.C.D.; Aranha, D.F. Faster unbalanced private set intersection. In Proceedings of the International Conference on Financial Cryptography and Data Security, Nieuwpoort, Curaçao, 26 February–2 March 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 203–221.
40. Pinkas, B.; Rosulek, M.; Trieu, N.; Yanai, A. PSI from PaXoS: Fast, malicious private set intersection. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020 ; pp. 739–767.
41. Garimella, G.; Pinkas, B.; Rosulek, M.; Trieu, N.; Yanai, A. Oblivious Key-Value Stores and Amplification for Private Set Intersection. In Proceedings of the CRYPTO 2021, Part II (LNCS 12826), Virtual, 16–20 August 2021; pp. 395–425.
42. Rindal, P.; Schoppmann, P. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, 4–8 May 2021; pp. 901–930.
43. Boneh, D.; Kogan, D.; Woo, K. Oblivious pseudorandom functions from isogenies. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Virtually, 7–11 December 2020; pp. 520–550.
44. Delpech de Saint Guilhem, C.; Pedersen, R. New proof systems and an OPRF from CSIDH. In Proceedings of the IACR International Conference on Public-Key Cryptography, Sydney, NSW, Australia, 15–17 April 2024; pp. 217–251.
45. Esgin, M.F.; Steinfeld, R.; Tairi, E.; Xu, J. LeOPaRd: Towards Practical Post-Quantum Oblivious PRFs via Interactive Lattice Problems. IACR ePrint 2024/1615. 2024. Available online: https://eprint.iacr.org/2024/1615.pdf (accessed on 14 December 2025).
46. Kaluderovic, N.; Cheng, N.; Mitrokotsa, K. A Post-Quantum Distributed OPRF from the Legendre PRF. IACR ePrint 2024/544. 2024. Available online: https://eprint.iacr.org/2024/544.pdf (accessed on 14 December 2025).
47. Ding, J.; Xie, X.; Lin, X. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem . Cryptology ePrint Archive, Report 2012/688. 2012. Available online: https://eprint.iacr.org/2012/688 (accessed on 14 December 2025).
48. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In *Advances in Cryptology—EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, France, 30 May–3 June 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 254–283.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.