Distance-Aware Private Set Intersection

Anrin Chakraborti
Duke University

Giulia Fanti
Carnegie Mellon University

Michael K. Reiter
Duke University

Abstract

Private set intersection (PSI) allows two mutually untrusting parties to compute an intersection of their sets, without revealing information about items that are not in the intersection. This work introduces a PSI variant called *distance-aware* PSI (DA-PSI) for sets whose elements lie in a metric space. DA-PSI returns pairs of items that are within a specified distance threshold of each other. This paper puts forward DA-PSI constructions for two metric spaces: (i) Minkowski distance of order 1 over the set of integers (i.e., for integers a and b, their distance is $|a-b|$); and (ii) Hamming distance over the set of binary strings of length ℓ. In the Minkowski DA-PSI protocol, the communication complexity scales logarithmically in the distance threshold and linearly in the set size. In the Hamming DA-PSI protocol, the communication volume scales quadratically in the distance threshold and is independent of the dimensionality of string length ℓ. Experimental results with real applications confirm that DA-PSI provides more effective matching at lower cost than naïve solutions.

1 Introduction

Private set intersection (PSI) is a widely-used multiparty cryptographic protocol, with applications across domains including contact discovery and tracing, private profile matching, privacy-preserving genomics, and collaborative learning. PSI protocols are used to compute the intersection (or common elements) of two or more sets held by mutually-untrusting parties. Critically, the parties learn no information about the elements that are *not* in the set intersection.

There is a long line of work on communication-efficient PSI protocols, with variants including different adversarial models [1, 11, 13, 16, 29]. However, these solutions are designed to return only *exact* matches. That is, an element appears in the intersection if and only if it matches (exactly) an element in each of the other parties' sets.

When exact matches may be rare, parties may want to privately compute *approximate* matches. For instance, given sets of points in Euclidean space, the intersection may contain all pairs within a certain Euclidean distance of each other. This notion has applications in domains where replacing an exact-matched set intersection with a distance-based intersection yields more effective systems:

- **Private collaborative blacklisting** enables mutually-untrusting parties to identify malicious network traffic and coordinated attacks. Typically, an intersection is computed privately over sets containing network identifiers, e.g., source IPs observed [22]. However, botnets usually span multiple (often contiguous) subnets [5, 38], and it is useful to compare ranges of IP addresses and detect overlaps.
- **Biometric identification** systems leverage Hamming distance/edit distance [7, 24, 36] and need to account for inexact/fuzzy matches due variations in sampling technologies. Fuzzy PSI functionalities have been used before in privacy-preserving biometric identification systems [32].
- **Credential stuffing identification** systems use PSI-like functionalities to detect password reuse across websites without revealing sensitive user information [34, 35]. These protocols consider exact password matches. However, it is useful to expand this idea for inexact/similar password matches, e.g., edit distance matches.

Distance-Aware Private Set Intersection (DA-PSI): In this paper we initiate the study of *distance-aware* private set intersection (DA-PSI). A DA-PSI protocol defined over a metric space allows two parties to compute an intersection of their respective sets containing all pairs of items that are within a predefined threshold distance in the metric space. Specifically, consider a metric space $(\mathcal{U}, \delta)$ with metric δ. Let the parties hold sets, A and B with n items each, where each item is a length-ℓ vector drawn from the space. The problem definition specifies a distance threshold d and requires the protocol to return $S \subseteq A \times B$ where $(\vec{a}, \vec{b}) \in S \Leftrightarrow \delta(\vec{a}, \vec{b}) \leq d$. A party learns no information about items in the counterparty's set that are not close to (within threshold of) one of its own elements.

Traditional PSI tools are not optimized for DA-PSI. A naïve application will need to check for all $\vec{a} \in A$ and for all $\vec{b} \in \mathcal{U}$ such that $\delta(\vec{a}, \vec{b}) \leq d$ if $\vec{b} \in B$. This is problematic since in many cases the search space is exponentially large. For example, Hamming distance with a distance threshold d will require searching over the Hamming ball of radius d around each $\vec{a} \in A$. There are over $\binom{\ell}{d} = \Omega((\frac{\ell}{d})^d)$ vectors around $\vec{a}$ in this Hamming ball. Thus, the communication cost of this protocol scales exponentially with the threshold and is impractical. This work poses the following question: *Can we design DA-PSI protocols where communication and compute costs scale polynomially in the distance threshold?*

We answer this question affirmatively by putting forward constructions for two important metric spaces: i) Hamming distance over the set $\{0,1\}^\ell$ for some fixed $\ell \in \mathbb{Z}^+$, and ii) Minkowski distance of order 1 over the set of integers (i.e., for integers a and b, their distance is $|a-b|$). In the following, we discuss the intuitions behind these protocols.

Hamming Distance-Aware PSI: Hamming distance is a good starting point since several other distances can be computed or approximated by Hamming distance [6, 12, 30]. We provide a construction building on the idea of sub-sampling each of a user's input vectors and mapping it to a unique set of sub-vectors such that the cardinality of the set difference between the sets corresponding to two vectors is exactly equal to the Hamming distance between the original vectors.

Our construction leverages additively homomorphic encryption and vector oblivious linear evaluation (VOLE). A key building block in the protocol is a novel sub-sampling mechanism which trades off accuracy (by allowing some false-positives) for better communication complexity: the communication cost scales polynomially in the distance threshold d and is *independent of the vector length* ℓ. Typically, for applications relying on Hamming distance comparisons, $d \ll \ell$ [20, 25], and thus the reduced set sizes after sub-sampling concretely improves communication costs over existing work [14, 25] . To compute the set differences, we propose a modified (and significantly simpler) version of the private set reconciliation protocol due to Ghosh and Simkin [11].

Integer distance-Aware PSI: We propose a DA-PSI protocol for Minkowski distance of first order over integers, loosely termed as the integer distance-aware PSI protocol. The communication cost scales linearly in the set size and logarithmically in the distance threshold; this is optimal with regards to the set sizes since linear communication is both necessary and sufficient for exact PSI [11, 17]. The key observation behind the protocol is that integers in a range $(a-d, a+d)$, where d is a specified distance threshold, can be succinctly represented by a collection of bit-strings corresponding to their binary representations. The total number of strings required is sublinear in d since multiple integers within a sequence will share prefixes, and the same common prefix will represent multiple consecutive integers. We design an algorithm to augment the inputs sets with $O(\log d)$ strings representing all integers in $(a-d, a+d)$. This mechanism is agnostic to the underlying cryptographic tools since any state-of-the-art PSI protocol can be augmented to provide an integer DA-PSI protocol.

	Comm	Computation			Dep	FPR, FNR
		Alice	Bob	Offline		
Hamming DA-PSI						
naïve	$O(n)$	$O\left(n\binom{\ell}{d}\right)$	$O(n)$	$O(1)$	–	–, –
Osadchy et al. [25]	$O(n^2\ell\lambda)$	$O(n^2)$	$O(n^2)$	$O(1)$	AHE	–, –
Huang et al. [15]	$O(n^2\ell\lambda)$	$O(n^2)$	$O(n^2)$	$O(1)$	OT	–, –
Uzun et al. [32]	$O\left(\frac{n^2T}{mB}\lambda\right)$	$O\left(\binom{T}{t}\left(\frac{nma}{T}\right)\right)$	$O\left(\frac{n^2T}{m}\right)$	$O\left(\frac{n^3T^2}{ma}\right)$	FHE	$0<\text{FPR}<1$ $0<\text{FNR}<1$
HamPSI (Sec. 4)	$O(n^2d^2\lambda)$	$O(n^2)$	$O(n^2)$	$O(1)$	AHE OLE	$0<\text{FPR}<1$ –
Integer DA-PSI						
naïve	$O(n\lambda d)$	$O(n)$	$O(n)$	$O(1)$	–	–, –
IntPSI (Sec. 5)	$O(n\lambda\log d)$	$O(n)$	$O(n)$	$O(1)$	–	–, –

Table 1: Asymptotic performance for our protocols in comparison to existing work. n: set size; d: distance threshold; ℓ: length of vectors; T,t: subsampling parameters, $T = O(\ell)$; m,a,B: FHE parameters, $m \gg T$. FPR, FNR $\in (0,1)$ are the false-positive and false-negative rates of the schemes; "–" indicates that the false positive or negative rate is negligible in λ. The schemes without dependencies are agnostic to the underlying primitives. The comm. cost of all existing Hamming DA-PSI protocol depends on the length of the vectors ℓ. The comm. cost of HamPSI is independent of ℓ.

Evaluation: We have implemented both protocols and benchmarked them on a public cloud. As an application of the integer distance-aware PSI, we have deployed it for collaborative blacklisting of IPs seen by real-world honeypots; for our parameter settings, the distance-aware PSI almost doubles the number of identified malicious IPs. For computing this intersection over sets containing roughly 25K IP addresses collected across all the honeypots, our protocol only requires 64 MB of communication and 1.5 seconds.

We have implemented our Hamming DA-PSI constructions. Micro-benchmarks show that it imposes 2-400× less communication than a generic garbled-circuit solution for distance thresholds up to 30. As an application, we have evaluated our protocol for the task of privately comparing vectors derived from iris images, and for a distance threshold sufficient to retrieve all of the matches in our dataset, it achieves 2.5× lower communication volume (with a false positive rate $\leq 10\%$ and no false negatives) than a generic secure 2PC baseline. When compared with the state-of-the-art Hamming containment query protocol by Uzun et al. [32], our protocol features 33-63% less communication and 33% less computation.

2 Related Work

Private set intersection is well-studied (e.g., [4, 8, 10, 18, 21, 26, 27, 28]). We refer to these for details on general PSI and focus on other distance-aware/fuzzy PSI primitives here.

Private Hamming Distance Computation: Table 1 compares our Hamming DA-PSI constructions with existing work on privately computing Hamming distance. Osadchy et al. [25] built a protocol using additively homomorphic encryption which enables a party to check when her vector is within a threshold Hamming distance of any of the vectors in a set held by the other party. A similar functionality is implemented by Huang et al. [14] using garbled circuits. For both protocols, the communication cost scales linearly in the vector sizes. In contrast, the cost of our Hamming distance aware protocol scales sublinearly in the vector size.

Uzun et al. [32] propose a protocol for Hamming distance comparisons over vectors derived from biometric identifiers. The protocol reduces the input vectors to sets of sub-vectors after a sub-sampling process. The sub-sampling protocol ensures that when two vectors are close, their corresponding sets have a certain number of matching elements. With these sets as inputs, the protocol implements a t-out-of-T matching protocol using fully homomorphic encryption (FHE), and leverages the ability of the FHE scheme to pack multiple ciphertexts using SIMD-style operations. In contrast, our constructions are based on computationally less-expensive primitives, namely additively homomorphic encryption (AHE) and oblivious linear evaluations (OLE).

3 Security Definitions & Background

Notation: $\mathbb{F}_p$ is a finite of field of prime order where p is a $O(\mathbf{poly}(\lambda))$-bit prime, and λ is a security parameter. $\mathbf{negl}(\cdot)$ is a function that is negligible in the input parameter; e.g., $\mathbf{negl}(\lambda) = O\left(2^{-\lambda}\right)$. $P(x) \in \mathbb{F}_p[x]$ is a polynomial with coefficients drawn from $\mathbb{F}_p$. The degree of polynomial $P(x)$ is represented by $\mathbf{deg}(P(x))$. The greatest common divisor of two (or more) polynomials is represented by $\mathbf{gcd}(P(x), Q(x))$. For polynomials $P(x)$ and $Q(x)$, $P_{/q}(x) := \frac{P(x)}{\mathbf{gcd}(P(x),Q(x))}$.
Rational Function: A rational function $r = \frac{P(x)}{Q(x)}$ has degree at most equal to the degree of the numerator + degree of the denominator. Let $V = \{(x_k, y_k)\}_{k=1}^{2t}$ be a set of points in $\mathbb{F}_p$. Then there exists a rational function with numerator and denominator in $\mathbb{F}_p[x]$ interpolating these points [19].
Parties: We assume that two *semi-honest* (a.k.a. honest-but-curious) mutually untrusting parties Alice and Bob run the protocols. The parties may learn information from the intermediate results but do not deviate from the protocol.
Distance-Aware Private Set Intersection: In this work, we are concerned with *distance-aware private set intersection* protocols, which we define here. Parties Alice and Bob are assumed to each store a set $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$, respectively, where the a_i's and b_i's are drawn from some universe $\mathcal{U}$, and $\delta : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ denotes a distance metric defined over $\mathcal{U}$. A distance-aware PSI protocol over metric space $(\mathcal{U}, \delta)$ with threshold d and input sets A, B returns a set $S \subseteq A \times B$ such that $S \triangleq \{(a,b) \ : \ a \in A, \ b \in B, \ \delta(a,b) \leq d\}$. We require this protocol to satisfy:

(1) **Correctness:** For any $(a,b) \in A \times B$,
- If $\delta(a,b) \leq d$, then $(a,b) \in S$ with probability $\geq$ TPR.
- If $\delta(a,b) > d$, then $(a,b) \notin S$ with probability $\geq$ TNR.

where TPR $\in (0,1)$ and TNR $\in (0,1)$ denote true positive and true negative rates respectively.

(2) **Security:** Alice learns only S and the cardinality of B, and Bob learns only S and the cardinality of A.

This definition allows for arbitrary false-positive and false-negative rates (i.e., FPR $= 1 -$ TNR and FNR $= 1 -$ TPR, respectively). This allows faster protocols (see Sec. 4) and accommodates protocols approximating one distance metric via another, e.g., with locality-sensitive hashing.

$\mathcal{F}_{ole}^{p}$: Ideal Functionality for Oblivious Linear Evaluation (OLE):

Parameters: Parties Alice and Bob, and finite field $\mathbb{F}_p$ from which inputs are drawn.

Inputs: Alice has input $x \in \mathbb{F}$ and Bob has as input a pair $(u, v) \in \mathbb{F}_p$.

Output: Alice learns $z = ux + v$. Bob learns $\perp$.

$\mathcal{F}_{vole}^{p}$: Ideal Functionality for Vector OLE (VOLE):

Parameters: Parties Alice and Bob, and finite field $\mathbb{F}_p$ from which inputs are drawn.

Inputs: Alice has input $x \in \mathbb{F}$ and Bob has as input a pair of vectors $(\vec{u}, \vec{v}) \in \mathbb{F}_p^{\ell} \times \mathbb{F}_p^{\ell}$.

Output: Alice learns $\vec{z} = \vec{u}x + \vec{v}$. Bob learns $\perp$.

Figure 1: Ideal functionalities for oblivious linear evaluation (OLE), and vector oblivious linear evaluation (VOLE)

3.1 Background

Oblivious Linear Evaluation (OLE) is a two-party cryptographic primitive wherein Alice inputs $x \in \mathbb{F}_p$; Bob inputs $u, v \in \mathbb{F}_p$; and Alice obtains $ux + v$ *without learning* u *and* v.
Vector Oblivious Linear Evaluation (VOLE) is an extension of the OLE functionality, where Bob's input is a pair of vectors, and Alice learns a linear combination of the vectors. Fig. 1 describes the VOLE functionality. The state-of-the-art VOLE protocol [37] is based on the learning parity with noise (LPN) assumption. The communication complexity of the protocol is linear in the vector length ℓ. Further technical details can be found in [37].
Threshold Set Intersection (a.k.a., t-out-T matching) is a variant of the PSI problem where the intersection of two (or more) sets is revealed to the parties if and only if the number of items in the intersection are above a certain predefined threshold. More formally, given two sets A and B of size n, and a threshold t, the protocol outputs S such that $S \triangleq A \cap B$ iff $|A \cap B| \geq n - t$. Otherwise, the protocol outputs $\perp$.

The state-of-the-art threshold PSI protocol (henceforth referred to a tPSI) is due to Ghosh and Simkin [11]. The main observation underlying the protocol is that given $A \setminus B$, the party holding A can obtain $A \cap B = A \setminus (A \setminus B)$. Thus, it suffices to design a threshold set reconciliation protocol where $A \setminus B$ (respectively, $B \setminus A$) is revealed to the parties iff $|A \setminus B| \leq t$. The protocol is inspired in part by the set reconciliation protocol due to Minsky et al. [23]; the idea is as follows: Alice and Bob encode the items of their corresponding sets, A and B in roots of polynomials $P(x)$ and $Q(x)$, respectively. If $|A \setminus B| \leq t$, then $\mathbf{deg}(\mathbf{gcd}(P(x), Q(x))) \geq n - t$, and so $r(x) = \frac{P(x)}{Q(x)}$ is a rational function of degree at most $2t$ (after cancellation of common roots in the numerator and denominator). $r(x)$ can be uniquely interpolated with $2t+1$ evaluation points. The denominator of $r(x)$ gives $A \setminus B$.

tPSI builds on this idea and tweaks the protocol to ensure that the elements in $B \setminus A$ are never revealed to Alice. Alice and Bob evaluate a polynomial $R_1(x)P(x) + R_2(x)Q(x)$ at $3t+1$ points, where $R_1(x)$ is a degree-t random polynomial contributed by Bob and $R_2(x)$ is a random degree-t polynomial contributed by Bob. Alice (and Bob) then compute the values of the rational function $r(x) = \frac{R_1(x)P(x)+R_2(x)Q(x)}{P(x)}$ at the aforementioned $3t+1$ points. Clearly, if $|A \setminus B| \leq t$, then $r(x)$ has a numerator of degree $\leq 2t$ and a denominator of degree $\leq t$ after cancellation of the common roots in $\mathbf{gcd}(P(x), Q(x))$ and $P(x)$. Since $r(x)$ is a rational functions of degree $\leq 3t$, it can be uniquely interpolated with the $3t+1$ evaluation points.

The security of the scheme relies on showing that the numerator of $r(x)$ after cancellation is a uniformly random polynomial. Specifically, let $R_1(x)P_{/q}(x) + R_2(x)Q_{/p}(x)$ be the numerator after canceling common roots in r. The following well-known result due to Kissner and Song [18] shows that this polynomial is uniformly random.

Lemma 1 ([18]). *Given two polynomials $P(x), Q(x) \in \mathbb{F}_p[x]$ with $\mathbf{deg}(P(x)) = \mathbf{deg}(Q(x)) \leq D_p$ such that $\mathbf{gcd}(P(x), Q(x))$ = 1, and two uniformly random polynomials, $R_1(x)$, $R_2(x)$ of degree $D_r \geq D_p$, the polynomial $R_1(x)P(x) + R_2(x)Q(x)$ is a uniformly random polynomial of degree $\leq D_r + D_p$.*

4 Protocol for Hamming Distances

We start with a protocol for privately computing a Hamming distance-aware set intersection between sets where elements are drawn from the universe $\mathcal{U} = \{0,1\}^{\ell}$. Fig. 2 defines the ideal functionality $\mathcal{F}^{h\text{-}PSI}_{\ell,d_{\mathsf{H}}}$ for Hamming DA-PSI between two parties with tunable true positive and true negative rates. We propose a protocol with $\mathrm{O}\left(n^2 \cdot \frac{d_{\mathsf{H}}^2}{\mathsf{FPR}} \cdot \lambda\right)$ communication cost for set sizes n (i.e., the cost is *independent of the vector length*), and compute time that scales polynomially in d_{H}.

Remark on Ideal Functionality: We have defined $\mathcal{F}^{h\text{-}PSI}_{\ell,d_{\mathsf{H}}}$ such that for each $(\vec{a}, \vec{b}) \in A \times B$, both parties learn $(\vec{a}, \vec{b})$ iff $\delta_H(\vec{a}, \vec{b}) \leq d_{\mathsf{H}}$. Another definition worth consideration is where Alice only learns if there is a $\vec{b} \in B$ such that $\delta_H(\vec{a}, \vec{b}) \leq d_{\mathsf{H}}$ *but not $\vec{b}$ itself.* However, this definition is not meaningful in the context of distance aware applications. For instance, there are $\binom{\ell}{d_{\mathsf{H}}}$ elements that are within Hamming distance d_{H} of an element $\vec{a} \in A$; it is not straightforward for Alice to guess $\vec{b}$ simply from the fact that $\delta_H(\vec{a}, \vec{b}) \leq d_{\mathsf{H}}$. This is unlike traditional PSI, where Alice can trivially guess Bob's element knowing that there is a match. Nonetheless, both our Hamming DA-PSI protocol have an additional Recover step where Alice obtains $\vec{b}$ from Bob after learning some intermediate results which indicates $\delta_H(\vec{a}, \vec{b}) \leq d_{\mathsf{H}}$. The protocol may be aborted at this stage (to save one extra round of communication) to realize an ideal functionality which only enables Alice to learn *if* $\delta_H(\vec{a}, \vec{b}) \leq d_{\mathsf{H}}$ without directly revealing $\vec{b}$.

$\mathcal{F}^{h\text{-}PSI}_{\ell,d_{\mathsf{H}}}$: Ideal Functionality for Hamming Aware DA-PSI:

Parameters: Parties Alice and Bob. Universe $\mathcal{U} = \{0,1\}^{\ell}$. Hamming distance threshold d_{H}. True positive rate TPR and true negative rate TNR.

Inputs: Alice has input $A = \{\vec{a}_i\}_{i=1}^{n} \subseteq \mathcal{U}$. Bob has input $B = \{\vec{b}_j\}_{j=1}^{n} \subseteq \mathcal{U}$.

Output: Alice and Bob learn $S \subseteq A \times B$ where for each $(\vec{a}, \vec{b}) \in A \times B$, if $\delta_H(\vec{a}, \vec{b}) \leq d_{\mathsf{H}}$ then $\mathbb{P}\left[(\vec{a}, \vec{b}) \in S\right] \geq$ TPR and if $\delta_H(\vec{a}, \vec{b}) > d_{\mathsf{H}}$ then $\mathbb{P}\left[(\vec{a}, \vec{b}) \notin S\right] \geq$ TNR.

$\mathcal{F}^{t\text{-}HQ}_{\ell,d_{\mathsf{H}}}$: Ideal Functionality for Threshold Hamming Query:

Parameters: Parties Alice and Bob. Universe $\mathcal{U} = \{0,1\}^{\ell}$. Hamming distance thresholds d_{H}, true positive rate TPR, and true negative rate TNR.

Inputs: Alice has vector $\vec{a} \in \mathcal{U}$ and Bob has vector $\vec{b} \in \mathcal{U}$.

Output: If $\delta_H(\vec{a}, \vec{b}) \leq d_{\mathsf{H}}$, then Alice and Bob learn $(\vec{a}, \vec{b})$ with probability $\geq$ TPR. If $\delta_H(\vec{a}, \vec{b}) > d_{\mathsf{H}}$, then Alice and Bob learn $\perp$ with probability $\geq$ TNR.

Figure 2: Ideal functionalities for Hamming distance-aware PSI and threshold Hamming queries

4.1 Threshold Hamming Query

The key building block of our construction is a protocol to privately determine if two bit vectors are within a certain Hamming distance of each other. We call this primitive a *threshold Hamming query*. Fig. 2 defines the ideal functionality $\mathcal{F}^{t\text{-}HQ}_{\ell,d_{\mathsf{H}}}$ for ℓ-bit vectors and Hamming distance threshold d_{H}.

4.1.1 tHamQueryLite: Hamming Query First Pass

We start with a simple and insecure version of our threshold Hamming query protocol dubbed tHamQueryLite (Fig. 3). The key observation is that $\mathcal{F}^{t\text{-}HQ}_{\ell,d_{\mathsf{H}}}$ can be realized as follows:

(1) **Map:** We use ℓ deterministic, injective mapping functions $M_1, \ldots M_\ell$ where $M_m : \{0,1\} \rightarrow \mathbb{F}_p$, $m \in [1, \ell]$, such that the ranges of the functions do not overlap. These functions map the individual bits in the vectors to elements of $\mathbb{F}_p$. The mth bit of vector $\vec{a}$, denoted $\vec{a}[m]$, is mapped to element $M_m(\vec{a}[m])$. $\vec{a}$ is then uniquely represented by $S_{\vec{a}} = \{M_1(\vec{a}[1]), \ldots, M_\ell(\vec{a}[\ell])\}$. Correspondingly, $\vec{b}$ is represented by $S_{\vec{b}} = \{M_1(\vec{b}[1]), \ldots, M_\ell(\vec{b}[\ell])\}$.

Parameters: Alice and Bob have vectors $\vec{a}, \vec{b} \in \{0,1\}^{\ell}$, respectively, and a Hamming distance threshold $d_{\mathsf{H}} \in [0, \ell/2)$.

Procedure Map**:**
Alice and Bob sample ℓ injective mapping functions $M_1, \ldots M_\ell$, $M_m : \{0,1\} \rightarrow \mathbb{F}_p$, $m \in [1, \ell]$. Alice computes set $S_{\vec{a}} := \{s_m : s_m := M_m(\vec{a}[m])\}$ and Bob computes $S_{\vec{b}} := \{s_m : s_m := M_m(\vec{b}[m])\}$.

Procedure OneSidedSetRecon**:**

(1) Alice and Bob select a set of $\ell + d_{\mathsf{H}} + 2$ points in $\mathbb{F}_p$ $X := \{x_k\}_{k=1}^{\ell+2d_{\mathsf{H}}+1}$ such that none of the points are in the ranges of any of the mapping functions.

(2) Alice encodes $S_{\vec{a}}$ in the polynomial $P(x) = \prod_{r \in S_{\vec{a}}} (x - r)$ and Bob encodes $S_{\vec{b}}$ in the polynomial $Q(x) = \prod_{r \in S_{\vec{b}}} (x - r)$.

(3) Bob samples two degree-ℓ random polynomials $R_1(x), R_2(x) \overset{\$}{:=} \mathbb{F}_p[x]$.

(4) For each $x_k \in X$, Alice sends $P(x_k)$ and Bob sends $R_1(x_k)$ and $R_2(x_k)Q(x_k)$ to $\mathcal{F}^p_{ole}$. Alice learns $W_A(x_k) := R_1(x_k)P(x_k) + R_2(x_k)Q(x_k)$ as the output of $\mathcal{F}^p_{ole}$.

(5) For $k \in [1, \ell + 2d_{\mathsf{H}} + 1]$, Alice computes the set of points

$$V := \{(x_k, y_k) : y_k := \tfrac{W_A(x_k)}{P(x_k)} = \tfrac{R_1(x_k)P(x_k) + R_2(x_k)Q(x_k)}{P(x_k)}\}.$$

(6) Alice interpolates V with a rational function $r(x) := \frac{\mathsf{Num}(x)}{\mathsf{Den}(x)}$ and checks that $\mathsf{Den}(x)$ is a factor of $P(x)$. If so, Alice outputs $S_{\vec{a}} \setminus S_{\vec{b}}$ which contains the roots of $\mathsf{Den}(x)$, otherwise Alice outputs $\perp$.

Procedure Recover**:**
If Alice receives $\perp$ from OneSidedSetRecon then output $\perp$. Otherwise, upon receiving $S_{\vec{a}} \setminus S_{\vec{b}}$: for each $s \in S_{\vec{a}} \cap S_{\vec{b}} = S_{\vec{a}} \setminus (S_{\vec{a}} \setminus S_{\vec{b}})$, if $s = M_m(\vec{a}[m])$ then $\vec{b}[m] = \vec{a}[m]$. For all indices $m' \in [1, \ell]$ that are left undetermined from $S_{\vec{a}} \setminus S_{\vec{b}}$, $\vec{b}[m'] = 1 - (\vec{a}[m'])$. Output $(\vec{a}, \vec{b})$.

Figure 3: tHamQueryLite: A first pass Hamming query protocol

(2) **Threshold Set Reconciliation:** A protocol with inputs $S_{\vec{a}}$ and $S_{\vec{b}}$ allows Alice to learn $S_{\vec{a}} \setminus S_{\vec{b}}$ iff $|S_{\vec{a}} \setminus S_{\vec{b}}| \leq d_{\mathsf{H}}$. Alice learns $\vec{b}$ from $S_{\vec{a}} \setminus S_{\vec{b}}$. E.g., let $\vec{a}$ = 1001, $\vec{b}$ = 1011, $S_{\vec{a}}$ = $\{M_1(1), M_2(0), M_3(0), M_4(1)\}$ and $S_{\vec{b}}$ = $\{M_1(1), M_2(0), M_3(1), M_4(1)\}$. Then, $S_{\vec{a}} \setminus S_{\vec{b}}$ = $\{M_3(0)\}$ and $\vec{b}[1] = \vec{a}[1]$, $\vec{b}[2] = \vec{a}[2]$, $\vec{b}[4] = \vec{a}[4]$ and $\vec{b}[3] = 1 - \vec{a}[3]$.

The mapping functions have no bearing on the security of the protocol, as long as the ranges do not overlap and the functions are injective. In our implementations we have used PRFs, but we do not rely on their *randomness* guarantees.

Threshold Set Reconciliation: In tHamQueryLite, we use OneSidedSetRecon, a new private set reconciliation protocol which is based on the tPSI protocol (see Sec. 3.1). OneSidedSetRecon allows one of the parties to learn $S_{\vec{a}} \setminus S_{\vec{b}}$ (say Alice) while the other party generates all the random coins. Both parties begin by encoding the items in their respective sets in the roots of polynomials $P(x)$ and $Q(x)$ respectively (see line 2 of OneSidedSetRecon in Fig. 3). This is followed by the parties jointly computing the evaluations of the polynomial $R_1(x)P(x) + R_2(x)Q(x)$ at $\ell + 2d_{\mathsf{H}} + 1$ points, where $R_1(x)$ and $R_2(x)$ are random polynomials sampled by Bob. This is achieved using $\ell + 2d_{\mathsf{H}} + 1$ calls to $\mathcal{F}^p_{ole}$ where Alice sends evaluations of $P(x)$ at each of the points and Bob correspondingly sends evaluations of

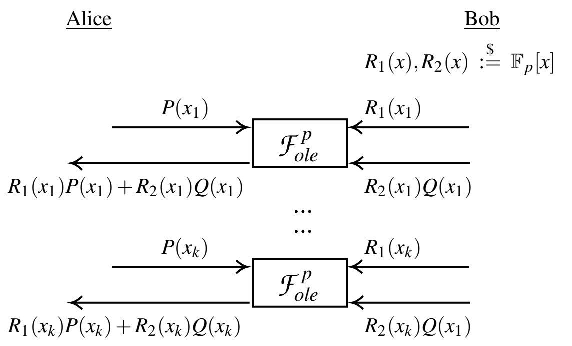


Figure 4: Using OLE for OneSidedSetRecon

$R_1(x)$, and $Q(x)R_2(x)$ (see Fig. 4). Finally, Alice interpolates the rational function $\frac{R_1(x)P(x) + R_2(x)Q(x)}{P(x)}$ with the evaluations of $R_1(x)P(x) + R_2(x)Q(x)$ similar to tPSI (line 6 of OneSidedSetRecon) and obtains $S_{\vec{a}} \setminus S_{\vec{b}}$ iff $|S_{\vec{a}} \setminus S_{\vec{b}}| \leq d_{\mathsf{H}}$. OneSidedSetRecon is simpler than tPSI with lower communication cost; OneSidedSetRecon requires only $\ell + 2d_{\mathsf{H}} + 1$ calls to $\mathcal{F}^p_{ole}$ compared to twice as many calls in tPSI while also avoiding two extra rounds of communication. The improvement comes from the fact that in contrast to tPSI where both parties learn the results, OneSidedSetRecon enables only Alice to learn the final result (see [3] for more details).

4.1.2 The (In)Security of tHamQueryLite

tHamQueryLite is not secure across all input parameters, and as we will show in this section, reveals $\vec{b}$ to Alice when $\delta_H(\vec{a}, \vec{b}) \in (d_{\mathsf{H}}, 2d_{\mathsf{H}})$. This is because OneSidedSetRecon reveals information when $|S_{\vec{a}} \setminus S_{\vec{b}}| \in (d_{\mathsf{H}}, 2d_{\mathsf{H}})$. In fact, the protocol of Ghosh and Simkin [11] on which OneSidedSetRecon is based also has the same leakage, and while the authors caution against using it as a standalone protocol[1], they have not analyzed this. More formally, we prove the following result.

Theorem 1. *Given sets $S_{\vec{a}}$ and $S_{\vec{b}}$ such that $|S_{\vec{a}}| = |S_{\vec{b}}|$ as inputs to* OneSidedSetRecon*, the following results hold:*

- ***Proposition 1:** If $|S_{\vec{a}} \setminus S_{\vec{b}}| \geq 2d_{\mathsf{H}}$, then there does not exist a PPT adversary that can determine any information regarding $S_{\vec{b}}$ from* OneSidedSetRecon *with more than negligible advantage (in λ) over guessing.*
- ***Proposition 2:** If $|S_{\vec{a}} \setminus S_{\vec{b}}| \in (d_{\mathsf{H}}, 2d_{\mathsf{H}})$, there exists an adversary that can determine $S_{\vec{b}}$ from* OneSidedSetRecon *with overwhelming probability (at least $1 - \mathbf{negl}(\lambda)$).*

Proof (sketch): The proof is in the full version of the paper [3]. Here, we provide the key arguments behind the proof.

Proof of Proposition 1: Consider the evaluation points Alice computes in line 5 corresponding to the rational function $\frac{R_1(x)P(x) + R_2(x)Q(x)}{P(x)} = \frac{\mathbf{gcd}(P(x),Q(x)) \times \left(R_1(x)P_{/q}(x) + R_2(x)Q_{/p}(x)\right)}{P(x)} =$

[1]To address the leakage, the paper proposes a significantly more expensive threshold cardinality of intersection protocol.

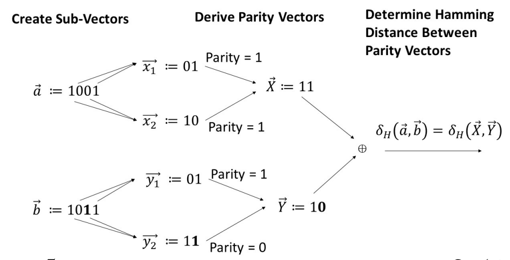


Figure 5: The process of computing $\delta_H(\vec{a},\vec{b})$ in tHamQueryRestricted. Alice and Bob partition the bits in $\vec{a}$ and $\vec{b}$ respectively into sub-vectors $\vec{x}_1, \vec{x}_2$ and $\vec{y}_1, \vec{y}_2$. They compute $\vec{X}$ and $\vec{Y}$ respectively using the parities of the sub-vectors. The bold bits show the locations where Alice's and Bob's inputs differ.

$\frac{R_1(x)P_{/q}(x)+R_2(x)Q_{/p}(x)}{P_{/q}(x)}$. Here, $P_{/q}(x) := \frac{P(x)}{\mathbf{gcd}(P(x),Q(x))}$. If the degree of $\mathbf{gcd}(P(x),Q(x)) = d_{\mathsf{GCD}}$, then $R_1(x)P_{/q}(x) + R_2(x)Q_{/p}(x)$ is a random polynomial of degree $2\ell - d_{\mathsf{GCD}}$. This is due to Lemma 1 as $P(x)$, $Q(x)$, $R_1(x)$, $R_2(x)$ are all degree-ℓ polynomials, and $R_1(x), R_2(x) \overset{\$}{:=} \mathbb{F}_p[x]$.

From the set of evaluation points, V, Alice may try to guess Bob's input polynomial $Q(x)$ and check whether there is a polynomial $\mathsf{Num}(x)$ of degree $2\ell - d_{\mathsf{GCD}}$ such that $\frac{\mathsf{Num}(x)}{P_{/q}(x)}$ is consistent with V. We show that when $|V| \leq 2\ell - d_{\mathsf{GCD}}$ which implies $\ell - d_{\mathsf{GCD}} \geq 2d_{\mathsf{H}}$, for every possible $P_{/q}(x)$ there is at least one candidate polynomial for $\mathsf{Num}(x)$. Since $R_1(x)P_{/q}(x) + R_2(x)Q_{/p}(x)$ is a random polynomial, any obtained value of $\mathsf{Num}(x)$ is equally likely to be $R_1(x)P_{/q}(x) + R_2(x)Q_{/p}(x)$. Moreover, if there are more than one candidates for $\mathsf{Num}(x)$, then they are all equally likely. Thus, Proposition 1 holds.

Proof of Proposition 2: When $|V| > 2\ell - d_{\mathsf{GCD}} + 1$ which implies $\ell - d_{\mathsf{GCD}} < 2d_{\mathsf{H}}$, the probability that Alice will find a candidate polynomial for $\mathsf{Num}(x)$ such that $\frac{\mathsf{Num}(x)}{P_{/q}(x)}$ is consistent with V *when she has incorrectly guessed* $Q(x)$ (and $P_{/q}(x)$) is negligible in λ. And so, Alice may check all possible candidates for $Q(x)$ and verify her guesses. The set of all possible values of $Q(x)$ is smaller than the set of degree-ℓ polynomials in $\mathbb{F}_p[x]$ since the roots of $Q(x)$ are fixed by the mapping functions $M_1, \ldots M_\ell$. There are 2^ℓ possible values of $Q(x)$, and for small ℓ, the search is computationally feasible for a PPT adversary. Thus, Proposition 2 holds. □

4.1.3 tHamQuery: Hamming Queries with Polynomial Computation

One way to fix tHamQueryLite is by checking if $\delta_H(\vec{a},\vec{b}) \in (d_{\mathsf{H}}, 2d_{\mathsf{H}})$; however, implementing this as a precursor to tHamQueryLite reveals more information than what the functionality allows when $\delta_H(\vec{a},\vec{b})$. We propose a protocol with *communication cost independent of the length of the vectors*, ℓ where the cases $\delta_H(\vec{a},\vec{b}) \in (d_{\mathsf{H}}, 2d_{\mathsf{H}})$ and $\delta_H(\vec{a},\vec{b}) \geq 2d_{\mathsf{H}}$ are indistinguishable.

Parameters: Alice and Bob have vectors $\vec{a}$ and $\vec{b}$ respectively, where $\vec{a},\vec{b} \in \{0,1\}^\ell$, $\delta_H(\vec{a},\vec{b}) \leq 2d_{\mathsf{H}}$ where d_{H} is the Hamming distance threshold. False positive rate $\textsc{fpr} \in (0, 0.5)$.

Procedure PermuteAndPartition:

(1) Alice and Bob sample a permutation $\pi : [1,\ell] \to [1,\ell]$ uniformly randomly from the set of all such permutations.

(2) Alice computes $\vec{a}_{\mathsf{perm}} \in \{0,1\}^\ell$ after permuting the bits of $\vec{a}$ as follows: for $m \in [1,\ell]$, $\vec{a}_{\mathsf{perm}}[\pi(m)] := \vec{a}[m]$. Similarly, Bob computes $\vec{b}_{\mathsf{perm}} \in \{0,1\}^\ell$ such that for $m \in [1,\ell]$ $\vec{b}_{\mathsf{perm}}[\pi(m)] := \vec{b}[m]$

(3) Alice creates $N_{\mathsf{bins}} = \frac{2d_{\mathsf{H}}^2}{\textsc{fpr}}$ sub-vectors $\vec{x}_1, \ldots, \vec{x}_{N_{\mathsf{bins}}}$ where each sub-vector is created by a contiguous sequence of $\frac{\ell}{N_{\mathsf{bins}}}$ bits of $\vec{a}_{\mathsf{perm}}$. Specifically, $\vec{x}_1 := \vec{a}_{\mathsf{perm}}[1 : \frac{\ell}{N_{\mathsf{bins}}}]$, $\vec{x}_2 := \vec{a}_{\mathsf{perm}}[\frac{\ell}{N_{\mathsf{bins}}} + 1 : 2 \times \frac{\ell}{N_{\mathsf{bins}}}], \ldots, \vec{x}_{N_{\mathsf{bins}}} := \vec{a}_{\mathsf{perm}}[(N_{\mathsf{bins}} - 1) \times \frac{\ell}{N_{\mathsf{bins}}} + 1 : \ell]$. Here $\vec{a}_{\mathsf{perm}}[m : m']$ is the contiguous sequence of bits starting from index m and up to (including) index m' in $\vec{a}_{\mathsf{perm}}$. Bob similarly creates $\vec{y}_1, \ldots, \vec{y}_{N_{\mathsf{bins}}}$ from $\vec{b}_{\mathsf{perm}}$.

(4) Alice computes vector $\vec{X} \in \{0,1\}^{N_{\mathsf{bins}}}$ such that for $m \in [1, N_{\mathsf{bins}}]$, $\vec{X}[m] := \mathbf{parity}(\vec{x}_m)$. Similarly, Bob computes vector $\vec{Y} \in \{0,1\}^{N_{\mathsf{bins}}}$ such that $\vec{Y}[m] := \mathbf{parity}(\vec{y}_m)$.

Protocol HamCompute:

(5) Alice generates a key-pair (pk, sk) for a semantically-secure additively homomorphic encryption $\mathsf{AHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and provides pk to Bob.

(6) For $m \in [1, N_{\mathsf{bins}}]$, Alice computes $ct_m := \mathsf{Enc}(\vec{X}[m])$ and $ct_w := \left\|\vec{X}\right\|$ where $\|.\|$ is the Hamming weight of the input vector. Alice sends $\{ct_w, ct_1, \ldots, ct_{N_{\mathsf{bins}}}\}$ to Bob.

(7) Bob computes $\mathsf{Enc}(\delta_H(\vec{X},\vec{Y})) := ct_w +_{pk} ct'_w -_{pk} 2 \times_{pk} \left((\vec{Y}[1] \times_{pk} ct'_1) +_{pk} \ldots +_{pk} (\vec{Y}[N_{\mathsf{bins}}] \times_{pk} ct'_{N_{\mathsf{bins}}})\right)$ where $ct'_w := \left\|\vec{Y}\right\|$, $+_{pk}$ $(-_{pk})$ is homomorphic addition (subtraction) of ciphertexts, and $\times_{pk}$ is scalar multiplication.

(8) Bob samples $\kappa \overset{\$}{:=} \mathbb{F}_p$ and computes $\mathsf{KeySet} := \{\kappa_i : \kappa_i := r_i \times_{pk} \left(\mathsf{Enc}(\delta_H(\vec{X},\vec{Y})) -_{pk} \mathsf{Enc}(i)\right) +_{pk} \mathsf{Enc}(\kappa),\ i \in [0, d_{\mathsf{H}}],\ r_i \overset{\$}{:=} \mathbb{F}_p\}$. Bob returns KeySet to Alice.

(9) Alice computes $\mathsf{KeySet}' := \{\kappa'_i : \kappa'_i := \mathsf{Dec}(\kappa_i),\ \kappa_i \in \mathsf{KeySet}\}$. If $\delta_H(\vec{a},\vec{b}) = \delta_H(\vec{X},\vec{Y}) \leq d_{\mathsf{H}}$, $\kappa \in \mathsf{KeySet}'$. Alice outputs KeySet'.

Figure 6: tHamQueryRestricted: Threshold Hamming query over restricted domain

Hamming Queries Over Restricted Domain: As the starting point, we present a protocol which distinguishes $\delta_H(\vec{a},\vec{b}) \leq d_{\mathsf{H}}$ and $\delta_H(\vec{a},\vec{b}) \in (d_{\mathsf{H}}, 2d_{\mathsf{H}}]$. The additional constraint is that for all inputs $\vec{a}$ and $\vec{b}$, the maximum Hamming distance between them is known apriori to be $\leq 2d_{\mathsf{H}}$. To generalize over the entire domain, we will subsequently extend this protocol and integrate with OneSidedSetRecon.

The protocol dubbed tHamQueryRestricted is inspired by a result due to Huang et al. [14]. The intuition is as follows: let S_I be the set of indices where $\vec{a}$ and $\vec{b}$ differ. By definition of the problem, $|S_I| \leq 2d_{\mathsf{H}}$. Consider the following balls and bins analysis: let the indices where $\vec{a}$ and $\vec{b}$ differ be represented by balls that are thrown randomly into $\frac{2d_{\mathsf{H}}^2}{\textsc{fpr}}$ empty bins, where $\textsc{fpr} \in (0, 0.5)$. Then, the following result shows that *all bins have* ≤ 1 *ball with probability at least* $1 - \textsc{fpr}$. It may be evident that the number of non-empty bins gives us $\delta_H(\vec{a},\vec{b})$.

Fact 1 ([14]). *If $2d_{\mathsf{H}}$ balls are randomly thrown into $\frac{2d_{\mathsf{H}}^2}{\text{FPR}}$ bins, where* FPR $\in (0, 0.5)$*, then with probability at most* FPR*, there is one or more bins with more than one ball.*

Fig. 6 describes the tHamQueryRestricted protocol built around this idea. The protocol comprises two procedures PermuteAndPartition and HamCompute. PermuteAndPartition uses a random permutation of the vectors to create $N_{\mathsf{bins}} = \frac{2d_{\mathsf{H}}^2}{\text{FPR}}$ sub-vectors. Specifically, the bits in $\vec{a}$ are partitioned into N_{bins} partitions (each corresponding to a sub-vector) after permuting with the random permutation. The resulting sub-vectors are denoted $\vec{x}_1, \ldots, \vec{x}_{N_{\mathsf{bins}}}$. Similarly, $\vec{b}$ is partitioned into $\vec{y}_1, \ldots, \vec{y}_{N_{\mathsf{bins}}}$ (lines 1–3). Then, Alice and Bob create parity vectors $\vec{X}$ and $\vec{Y}$ using the parities of the sub-vectors (line 4).

HamCompute privately computes the Hamming distance between the parity vectors and compares it with the distance threshold. Alice sends $\vec{X}$ to Bob, with each bit encrypted individually (line 6). Bob computes $\mathsf{Enc}(\delta_H(\vec{X}, \vec{Y}))$ by computing the Hamming distance over encrypted bits (line 7). The encryption scheme used is additively homomorphic, which ensures that Bob can compute over the encrypted bits. This gives $\mathsf{Enc}(\delta_H(\vec{a}, \vec{b}))$ due to following fact: since each pair $\vec{x}_i, \vec{y}_i$ can differ in at most one bit due to Fact 1, $\delta_H(\vec{x}_i, \vec{y}_i) = \delta_H(\mathbf{parity}(\vec{x}_i), \mathbf{parity}(\vec{y}_i))$. Then, $\delta_H(\vec{a}, \vec{b}) = \sum_{i=1}^{N_{\mathsf{bins}}} \delta_H(\mathbf{parity}(\vec{x}_i), \mathbf{parity}(\vec{y}_i)) = \delta_H(\vec{X}, \vec{Y})$ (see Fig. 5).

Finally, Bob samples a key $\kappa \overset{\$}{:=} \mathbb{F}_p$ and returns a set KeySet containing κ "blinded" by random values, and available to Alice iff. $\delta_H(\vec{X}, \vec{Y}) \in [0, d_{\mathsf{H}}]$. Specifically, for each $i \in [0, d_{\mathsf{H}}]$, Bob computes $\mathsf{Enc}(r_i \times (\delta_H(\vec{X}, \vec{Y}) - i) + \kappa)$ using the homomorphic properties of the encryption scheme and returns the encrypted values obtained in the KeySet to Alice where $r_i \overset{\$}{:=} \mathbb{F}_p$. Alice obtains κ only when $\delta_H(\vec{X}, \vec{Y}) \in [0, d_{\mathsf{H}}]$ (lines 8–9). Otherwise, $r_i \times (\delta_H(\vec{X}, \vec{Y}) - i) + \kappa \overset{\$}{:=} \mathbb{F}_p$.

General Hamming Queries: We are ready to combine tHamQueryRestricted and OneSidedSetRecon to achieve a secure threshold Hamming query protocol, tHamQuery. The protocol requires a PRF over a finite field $\phi : \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{F}_p$. The outline of this integration is (see Fig. 7):

(1) Alice and Bob send $\vec{a}$ and $\vec{b}$ to tHamQueryRestricted respectively. Alice obtains KeySet from which she can obtain κ only when $\delta_H(\vec{a}, \vec{b}) \notin (d_{\mathsf{H}}, 2d_{\mathsf{H}}]$.

(2) Alice builds set $S'_{\vec{a}} := \{\vec{x}_1, \ldots, \vec{x}_{N_{\mathsf{bins}}}\}$ where $\vec{x}_1, \ldots, \vec{x}_{N_{\mathsf{bins}}}$ are the sub-vectors created during PermuteAndPartition in tHamQueryRestricted (see line 3 of Fig. 6). Bob builds $S'_{\vec{b}} := \{\vec{y}_1, \ldots, \vec{y}_{N_{\mathsf{bins}}}\}$.

(3) Alice and Bob run OneSidedSetRecon with $S'_{\vec{a}}$ and $S'_{\vec{b}}$ as inputs and threshold d_{H} with one change: Bob modifies his inputs to $\mathcal{F}^p_{ole}$ such that Alice obtains a "blinded" set of evaluations, i.e., for $k \in [1, N_{\mathsf{bins}} + 2d_{\mathsf{H}} + 1]$, Alice obtains $R_1(x_k)P(x_k) + R_2(x_k)Q(x_k) + \phi(\kappa, k)$ (see Fig. 8).

<u>**Parameters:**</u> Alice and Bob have vectors $\vec{a}, \vec{b} \in \{0,1\}^\ell$ respectively. and a Hamming distance threshold $d_{\mathsf{H}} \in [0, \ell/2)$.

<u>**Procedure** PermuteAndPartition**:**</u>
Alice and Bob run tHamQueryRestricted with $\vec{a}$ and $\vec{b}$ as inputs. Alice obtains $S'_{\vec{a}} := \{\vec{x}_1, \ldots, \vec{x}_{N_{\mathsf{bins}}}\}$ and KeySet, while Bob obtains $S'_{\vec{b}} := \{\vec{y}_1, \ldots, \vec{y}_{N_{\mathsf{bins}}}\}$ where $N_{\mathsf{bins}} = \frac{2d_{\mathsf{H}}^2}{\text{FPR}}$.

<u>**Procedure** OneSidedSetReconBlind**:**</u>

(1) Alice and Bob select a set of $N_{\mathsf{bins}} + 2d_{\mathsf{H}} + 1$ points in $\mathbb{F}_p$, $X := \{x_k\}_{k=1}^{N_{\mathsf{bins}}+2d_{\mathsf{H}}+1}$ such that none of the points are in $S'_{\vec{a}}$ and $S'_{\vec{b}}$.

(2) Alice and Bob compute $P(x) := \prod_{r \in S'_{\vec{a}}} (x - r)$ and $Q(x) := \prod_{r \in S'_{\vec{b}}} (x - r)$ respectively. Bob samples two random polynomials $R_1(x), R_2(x) \overset{\$}{:=} \mathbb{F}_p[x]$ of degree N_{bins}.

(3) For each $x_k \in X$, Alice sends $P(x_k)$ to $\mathcal{F}^p_{ole}$, while Bob sends $R_1(x_k)$, and $R_2(x_k) \times Q(x_k) + \phi(\kappa, k)$. Alice obtains $W_A(x_k) := R_1(x_k)P(x_k) + R_2(x_k)Q(x_k) + \phi(\kappa, k)$

(4) For each element $\kappa_i \in$ KeySet, Alice obtains a candidate key, $\kappa'_i := \mathsf{Dec}(\kappa_i)$. Alice computes
$$V_i := \{(x_k, y_k) : y_k := \tfrac{W_A(x_k) - \phi(\kappa'_i, k)}{P(x_k)}\}.$$

(5) For each $i \in [0, d_{\mathsf{H}}]$, Alice interpolates V_i with a rational function $r(x) := \frac{\mathsf{Num}(x)}{\mathsf{Den}(x)}$ and checks that $\mathsf{Den}(x)$ is a factor of $P(x)$. If so, Alice outputs $S'_{\vec{a}} \setminus S'_{\vec{b}}$ which contains the roots of $\mathsf{Den}(x)$, otherwise Alice outputs $\perp$.

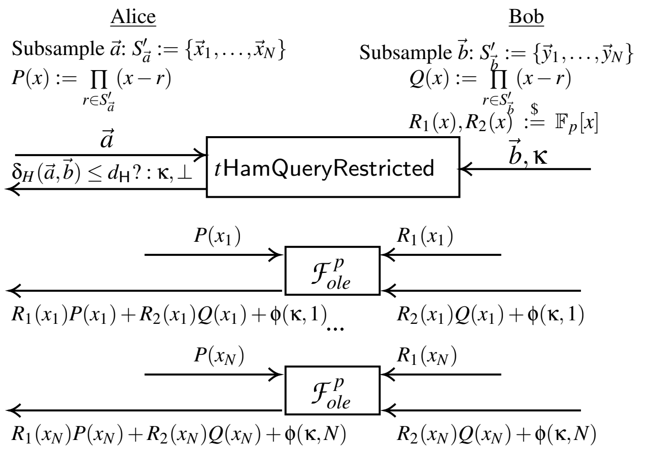
<u>**Procedure** Recover**:**</u>
If Alice receives $\perp$ from OneSidedSetReconBlind then output $\perp$. Otherwise, upon receiving $S'_{\vec{a}} \setminus S'_{\vec{b}}$, obtain $\vec{b}$ from Bob. Output $(\vec{a}, \vec{b})$.

Figure 7: tHamQuery: Threshold Hamming query protocol

Theorem 2. *Assuming that there exists a semantically-secure additively homomorphic encryption scheme that produces* $O(\lambda)$*-bit ciphertexts, and that there is a protocol for* $\mathcal{F}^p_{ole}$ *that requires* $O(\lambda)$ *bits of communication, for false positive rate* FPR $\in (0, 0.5)$*,* tHamQuery *realizes* $\mathcal{F}^{t\text{-}HQ}_{\ell, d_{\mathsf{H}}}$ *with* $O\left(\frac{d_{\mathsf{H}}^2}{\text{FPR}} \cdot \lambda\right)$ *bits of communication and compute costs polynomial in* d_{H}.

Proof (sketch): The communication cost of the protocol is straightforward. Alice sends $|\vec{X}| = \frac{2d_{\mathsf{H}}^2}{\text{FPR}}$ encrypted bits to Bob. Bob sends back the encrypted KeySet with $|\mathsf{KeySet}| = d_{\mathsf{H}}$ ciphertexts. Finally, there are $\frac{2d_{\mathsf{H}}^2}{\text{FPR}} + 2d_{\mathsf{H}} + 1$ calls to $\mathcal{F}^p_{ole}$, each of which requires $O(\lambda)$ bits of communication.

The following arguments show that the protocol is secure. Alice can "unblind" and obtain the correct evaluations of $R_1(x)P(x) + R_2(x)Q(x)$ iff she has obtained κ in Step 1, which happens with high probability (at least $1 - \varepsilon$) when $\delta_H(\vec{a}, \vec{b}) \notin (d_{\mathsf{H}}, 2d_{\mathsf{H}}]$. Otherwise, Alice obtains random points as evaluations of $R_1(x)P(x) + R_2(x)Q(x)$ which reveals no information regarding $S'_{\vec{b}}$. If $\delta_H(\vec{a}, \vec{b}) > 2d_{\mathsf{H}}$, Alice may still obtain $\kappa \in$ KeySet since Fact 1 is applicable only when $\delta_H(\vec{a}, \vec{b}) \leq 2d_{\mathsf{H}}$. We show in the full proof [3] that $|S'_{\vec{a}} \setminus S'_{\vec{b}}| \geq 2d_{\mathsf{H}}$ with high probability (at least $1 - \varepsilon$), and as Theorem 1 shows, when $|S'_{\vec{a}} \setminus S'_{\vec{b}}| \geq 2d_{\mathsf{H}}$, Alice learns nothing about $S'_{\vec{a}} \setminus S'_{\vec{b}}$ from the evaluations of $R_1(x)P(x) + R_2(x)Q(x)$. □


Alice
Bob
Subsample $\vec{a}$: $S'_{\vec{a}} := \{\vec{x}_1, \ldots, \vec{x}_N\}$
$P(x) := \prod_{r \in S'_{\vec{a}}} (x - r)$
Subsample $\vec{b}$: $S'_{\vec{b}} := \{\vec{y}_1, \ldots, \vec{y}_N\}$
$Q(x) := \prod_{r \in S'_{\vec{b}}} (x - r)$
$R_1(x), R_2(x) \overset{\$}{:=} \mathbb{F}_p[x]$
$\vec{a}$
$\delta_H(\vec{a}, \vec{b}) \le d_{\mathsf{H}} ? : \kappa, \perp$
tHamQueryRestricted
$\vec{b}, \kappa$
$P(x_1)$
$R_1(x_1)$
$\mathcal{F}^p_{ole}$
$R_1(x_1)P(x_1) + R_2(x_1)Q(x_1) + \phi(\kappa, 1)$
$R_2(x_1)Q(x_1) + \phi(\kappa, 1)$
...
$P(x_N)$
$R_1(x_N)$
$\mathcal{F}^p_{ole}$
$R_1(x_N)P(x_N) + R_2(x_N)Q(x_N) + \phi(\kappa, N)$
$R_2(x_N)Q(x_N) + \phi(\kappa, N)$
Figure 8: Using OLE and tHamQueryRestricted in tHamQuery

4.2 HamPSI: Hamming DA-PSI from tHamQuery

Building a Hamming DA-PSI protocol based on the Hamming query mechanism described so far is straightforward. Let $A := \{\vec{a}_1, \ldots, \vec{a}_n\}$ and $B := \{\vec{b}_1, \ldots, \vec{b}_n\}$ be Alice's and Bob's inputs to the Hamming DA-PSI protocol. Then, for $(i, j) \in n \times n$, Alice and Bob run tHamQuery with $\vec{a}_i$ and $\vec{b}_j$ as inputs.

We present the full protocol, denoted HamPSI, in App. B with a further optimization using vector OLE (see Sec. 3.1) This optimization improves communication costs and compute times without impacting security.

Theorem 3. *Assuming that there exists a semantically secure additively homomorphic encryption scheme, and a protocol securely realizing $\mathcal{F}^p_{vole}$ with $O(n\lambda)$ bits of communication, there is a Hamming DA-PSI protocol which securely realizes $\mathcal{F}^{h\text{-}PSI}_{\ell,d_{\mathsf{H}}}$ with $O\left(n^2 \cdot \frac{d_{\mathsf{H}}^2}{\text{FPR}} \cdot \lambda\right)$ bits of communication where* FPR $\in (0, 0.5)$ *is the false positive rate.*

4.3 HamPSISample: Sub-Sampling Based Hamming DA-PSI

So far we have discussed a way to build a Hamming DA-PSI protocol using tHamQuery which fixes tHamQueryLite by explicitly checking if the inputs, $\delta_H(\vec{a}, \vec{b}) \in (d_{\mathsf{H}}, 2d_{\mathsf{H}})$. However, there is an alternate way to fix the problem which leads to a more communication-efficient protocol, but at the cost of additional computation. The protocol uses a Hamming query protocol, denoted tHamQueryExp, which relies only on OLE. tHamQueryExp: The protocol is based on the findings of Theorem 1. Specifically, as the proof shows when Alice and Bob interpolate $R_1(x)P(x) + R_2(x)Q(x)$ at $|V| > 2\ell - d_{\mathsf{GCD}} + 1$ points in OneSidedSetRecon, Alice can retrieve $S_{\vec{a}} \setminus S_{\vec{b}}$ when $|S_{\vec{a}} \setminus S_{\vec{b}}| = \ell - d_{\mathsf{GCD}}$ with overwhelming probability (Proposition 2). On the other hand, when $|S_{\vec{a}} \setminus S_{\vec{b}}| > \ell - d_{\mathsf{GCD}}$, the evaluation points reveal nothing to Alice (Proposition 1).

Parameters: Alice and Bob have vectors $\vec{a}, \vec{b} \in \{0,1\}^\ell$, respectively, and a Hamming distance threshold d_{H}.
Procedure Map: Follows the steps of tHamQueryLite.

Procedure OneSidedSetReconExp:
(1) Alice and Bob select a set of $\ell + d_{\mathsf{H}} + 2$ points in $\mathbb{F}_p$, $X := \{x_k\}_{k=1}^{\ell+d_{\mathsf{H}}+2}$ such that none of the points are in $S_{\vec{a}}$ and $S_{\vec{b}}$.
(2) Alice encodes $S_{\vec{a}}$ in the polynomial $P(x) = \prod_{r \in S_{\vec{a}}} (x - r)$ and Bob encodes $S_{\vec{b}}$ in the polynomial $Q(x) = \prod_{r \in S_{\vec{b}}} (x - r)$. Bob samples two degree-ℓ random polynomials $R_1(x), R_2(x) \overset{\$}{:=} \mathbb{F}_p[x]$.
(3) For each $x_k \in X$, Alice sends $P(x_k)$ and Bob sends $R_1(x_k)$ and $R_2(x_k) \times Q(x_k)$ to $\mathcal{F}^p_{ole}$. Alice learns $W_A(x_k) := R_1(x_k)P(x_k) + R_2(x_k)Q(x_k)$ as the output of $\mathcal{F}^p_{ole}$.
(4) Alice computes and interpolates V_0 with a rational function of degree $\ell + d_{\mathsf{H}}$ similar to line 6 of OneSidedSetRecon .
$$V_0 := \{(x_k, y_k) : y_k := \tfrac{W_A(x_k)}{P(x_k)},\ k \in [1, \ell + d_{\mathsf{H}} + 1]\}.$$
(5) If Alice outputs $S_{\vec{a}} \setminus S_{\vec{b}}$ from the step above, then output $S_{\vec{a}} \setminus S_{\vec{b}}$ and abort. Otherwise, for each $t \in (d_{\mathsf{H}}/2, d_{\mathsf{H}}]$, Alice computes all possible values of $S_{\vec{a}} \setminus S_{\vec{b}}$ such that $|S_{\vec{a}} \setminus S_{\vec{b}}| = t$. Let C_{sub} be the set of all such sets across all $t \in (d_{\mathsf{H}}/2, d_{\mathsf{H}}]$.
(6) For each $C_i \in C_{\mathsf{sub}}$, Alice computes the polynomial $P_i(x) := \prod_{r \in C_i} (x - r)$ and computes the set of points
$$V_i := \{(x_k, y_k) : y_k := \tfrac{W_A(x_k)}{P_i(x_k)},\ k \in [1, \ell + d_{\mathsf{H}} + 2]\}.$$
(7) Alice interpolates V_i for each $C_i \in C_{\mathsf{sub}}$ with a polynomial. If the degree of the interpolating polynomial is $\le \ell + \mathbf{deg}(P_i(x))$, Alice outputs $S_{\vec{a}} \setminus S_{\vec{b}} := C_i$.
Procedure Recover: Follows the steps of tHamQueryLite

Figure 9: tHamQueryExp: Hamming queries with exponential compute costs.

So, one way to fix OneSidedSetRecon is by evaluating $R_1(x)P(x) + R_2(x)Q(x)$ at $(2\ell - d_{\mathsf{GCD}} + 1) + 1 = (2\ell - (\ell - d_{\mathsf{H}}) + 1) + 1 = \ell + d_{\mathsf{H}} + 2$ points. In this way, Alice learns $S_{\vec{a}} \setminus S_{\vec{b}}$ with overwhelming probability when $|S_{\vec{a}} \setminus S_{\vec{b}}| \le d_{\mathsf{H}}$ but nothing otherwise. The modified protocol is called OneSidedSetReconExp (Fig. 9). Similar to OneSidedSetRecon, Alice and Bob compute polynomials $P(x)$ and $Q(x)$ from their respective sets. Then, they evaluate $R_1(x)P(x) + R_2(x)Q(x)$ at $\ell + d_{\mathsf{H}} + 2$ points using calls to $\mathcal{F}^p_{ole}$. Alice first attempts to interpolate $\ell + d_{\mathsf{H}} + 1$ points with a rational function of degree $\ell + d_{\mathsf{H}}$ (lines 2–4). Note if $|S_{\vec{a}} \setminus S_{\vec{b}}| \le d_{\mathsf{H}}/2$, then this step will reveal $S_{\vec{a}} \setminus S_{\vec{b}}$ to Alice.

Otherwise, Alice computes for each $t \in (d_{\mathsf{H}}/2, d_{\mathsf{H}}]$, each possible value of $S_{\vec{a}} \setminus S_{\vec{b}}$ such that $|S_{\vec{a}} \setminus S_{\vec{b}}| = t$ (line 5). Let C_{sub} be the set of all such sets. Then, for each $C_i \in C_{\mathsf{sub}}$, Alice computes $P_i(x) := \prod_{r \in C_i} (x - r)$. Finally, Alice checks if a polynomial $\mathsf{Num}(x)$ of degree $\le \ell + \mathbf{deg}(P_i(x))$ exists such that $\frac{\mathsf{Num}(x)}{P_i(x)}$ is consistent with the points obtained for the rational function $\frac{R_1(x)P(x) + R_2(x)Q(x)}{P_i(x)}$ (line 7). Due to Proposition 2, there is a negligible probability of obtaining a false positive, i.e., Alice finds $\mathsf{Num}(x)$ when her guess for $S_{\vec{a}} \setminus S_{\vec{b}}$ is incorrect. There are no false negatives.
Reducing Search Space by Sub-Sampling: The total search

Parameters: Parties Alice and Bob. Integer distance threshold $d \geq 0$.

Inputs: Alice has input $A = \{a_i : a_i \in \mathbb{Z}^+ \cup \{0\}\}$. Bob has input $B = \{b_j : b_j \in \mathbb{Z}^+ \cup \{0\}\}$. $|A| = |B| = n$.

Output: Alice and Bob learn $S \subseteq A \times B$ where if $(a,b) \in A \times B, |a-b| \leq d_{\text{int}}$ then $\mathbb{P}[(a,b) \in S] \geq$ TPR and if $|a-b| > d_{\text{int}}$ then $\mathbb{P}[(a,b) \notin S] \geq$ TNR.

Figure 10: Ideal functionality $\mathcal{F}^{i\text{-}PSI}_{d_{\text{int}}}$ for Integer DA-PSI.

space for this process is over $\sum_{t=d_H/2+1}^{d_H} \binom{\ell}{t}$ guesses for $S_{\vec{a}} \setminus S_{\vec{b}}$, and is not feasible with large vectors and distance thresholds. However, as we will show in Sec. 6, when we replace Map with a sub-sampling algorithm [2, 9, 32, 33] reducing large vectors to a small set of sub-vectors, this method outperforms the existing state of the art [32]. The cost savings come from the fact that our protocol only relies on cheap symmetric-key primitives while the protocol of Uzun et al. [32] relies on fully homomorphic encryption. Based on this idea, we have built and implemented a DA-PSI protocol combining the sub-sampling algorithm from [2, 9, 32, 33] with OneSidedSetReconExp, denoted HamPSISample. More details of the protocol are presented in App. C.

5 Protocol for Integer Distances

In this section, we present a DA-PSI protocol for L_1 distance of order 1 over integers, loosely termed as integer distance-aware PSI. The protocol requires $O(n\lambda \log d_{\text{int}})$ bits of communication for computing the intersection of two sets of size n where d_{int} is a user-specified distance threshold.

Ideal Functionality: The ideal functionality for an integer distance-aware PSI is defined in Fig. 10. Note that $(a_i, b_j) \in S$ only when $b_j \in (a_i - d_{\text{int}}, a_i + d_{\text{int}})$. The range excludes the boundary elements, $a_i + d_{\text{int}}$ and $a_i - d_{\text{int}}$. This is primarily for ease of description of the protocol and it is trivial to extend the functionality and the protocol to include the boundary elements. Also, while the functionality allows tunable true positive and true negative rates, *the protocol we present is correct with probability* 1.0*, i.e.,* TPR = TNR = 1.

Observe that an inefficient realization of $\mathcal{F}^{i\text{-}PSI}_{d_{\text{int}}}$ immediately exists: Alice creates an augmented set, $\hat{A}$ with all integers $(a - d_{\text{int}}, a + d_{\text{int}})$ for each $a \in A$. Any generic PSI protocol can be used for computing the intersection between the augmented set and B. This protocol however requires $O(n\lambda d_{\text{int}})$ bits of communication when using a PSI protocol with communication cost scaling linearly in the set size.

Key Idea: To reduce overall communication, we will reduce the number of items in the augmented set. The key observation behind this reduction is that all integers in the neighborhood of an integer $a \in A$, $a', |a' - a| \leq d_{\text{int}}$ can be succinctly represented by a collection of bit strings corresponding to their binary representations. The total number of such strings required is sublinear in d_{int} since multiple integers within a sequence will share prefixes, and the same common prefix can be used to represent multiple consecutive integers. For instance, the binary representation of 42 (101010) and 43 (101011) share the prefix 10101. Both these integers can be represented by the string 10101$*$ where $*$ denotes a wildcard bit. Leveraging this fact, the idea is to generate the least number of bit strings to represent all integers $a', |a' - a| \leq d_{\text{int}}$. The problem is reduced to string matching over these bit strings.

The augmenting process is discussed next. The protocol we will present allows one of the parties, say Alice, to learn the distance-aware intersection, and then this information can be shared with Bob using an extra round of communication.

Augmenting Alice's Set: The augmented set $\hat{A}$ includes fixed-length strings representing $(a - d_{\text{int}}, a + d_{\text{int}})$ for each $a \in A$. These strings are obtained from the prefixes of fixed-length binary representations of all integers in the range. This fixed length, denoted MaxBitLen, may be determined from the universe from which the elements in A and B are drawn.

Intuitively, the process is based on two observations. First, the integers in $(a - d_{\text{int}}, a + d_{\text{int}})$ can differ only in their $\lfloor \log(2d_{\text{int}} - 1) \rfloor + 1$ least significant bits if $2^k \leq a - d \leq a + d \leq 2^{k+1}$. Second, the MaxBitLen-sized binary representations of all integers in $[2^k, 2^{k+1} - 1], k \in \mathbb{Z}^+$ have a common prefix of length MaxBitLen $- k - 1$. So, all these integers can be represented by a string formed by appending $k+1$ wildcard bits to the common prefix. Based on these observations, the idea is to recursively partition $(a - d_{\text{int}}, a + d_{\text{int}})$ into smaller ranges of the form $[0, 2^k - 1]$ or $[2^k, 2^{k+1} - 1]$ and obtain a *representative string* for each such range. More formally, to create representative strings for integers in $(a - d_{\text{int}}, a + d_{\text{int}})$, we identify the *enclosing common* prefixes.

Definition 1. *Given any arbitrary set of bit strings, an enclosing common prefix of length $\ell_p \leq$ MaxBitLen satisfies:*

(1) There are $2^{(\text{MaxBitLen} - \ell_p)}$ bit strings which have this prefix in common.

(2) The bit strings which share this prefix do not have a common prefix of length $< \ell_p$.

All identified enclosing common prefixes are appended with wildcard bits to generate representative strings. We show later that for each $a \in A$ the number of enclosing common prefixes is $O(\log d_{\text{int}})$. Intuitively this is because the range of integers is recursively halved and each such range has a constant number of enclosing common prefixes. Thus, the augmented set contains $O(n \cdot \log d_{\text{int}})$ representative strings.

Example: We are interested in the representative strings for all integers in the range (41, 56) (see Fig. 11). The 8-bit binary representation of 42 is 00101010 and the 8-bit binary representation of 55 is 00110111. Integers in the range [42, 43] have a common enclosing prefix 0010101, integers in the range [44, 47] have a common enclosing prefix 001011 and the integers in the range [48, 55] have a common enclosing

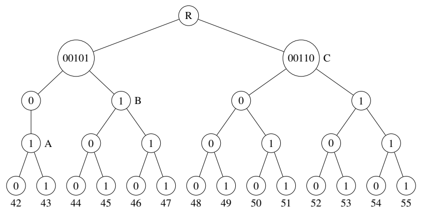


Figure 11: Prefix trie examples. (a) Trie built over the binary representations of integers [42, 55]. The nodes marked A, B and C are the roots of the *maximal enclosing complete* subtries.

prefix 00110. Thus, 8-bit representative strings for all integers in $(41,56)$ are $0010101*$, $001011**$ and $00110***$.

Augmenting Bob's Set: To check whether an integer falls in any of the ranges in an augmented set, we need to check whether it shares a common prefix with any of the representative strings. The augmented set $\hat{B}$ includes these strings. Specifically we need to check prefixes only of length $\ell_p \in [\mathsf{MaxBitLen} - \lfloor \log(2d-1) \rfloor - 1, \mathsf{MaxBitLen}]$ (as will be discussed later). Thus, representative strings for each integer $b_j \in B$ is obtained by replacing the required number of least significant bits in the binary representation of b_j with wildcard bits. Specifically, the first representative string is generated by replacing the least significant bit, the second string is generated by replacing the last two least significant bits and so on. All the representative strings for all $b_j \in B$ is part of $\hat{B}$. Thus, $|\hat{B}| = \mathrm{O}(n \cdot \log d_{\mathsf{int}})$.

Example: Consider $b = 49$ (00110001). The 8-bit representative strings of 49 are 00110001, $0011000*$, $001100**$, $00110***$, $0011****$. The set of strings for integers in range $(41,56)$ and the strings for 49 have the string $00110***$ in common which correctly shows that $b \in (41,56)$.

5.1 Technical Description

Algorithm for Augmenting Sets: The algorithm (Algorithm 1) has two procedures corresponding to the processes of augmenting Alice's input A and Bob's input B. To generate representative strings for each $a_i \in A$, the algorithm first builds a prefix trie over the bit strings corresponding to the binary representations (of length $\mathsf{MaxBitLen}$) of integers in $(a_i - d_{\mathsf{int}}, a_i + d_{\mathsf{int}})$ (Steps 8 - 11). As usual, each bit string corresponds to a path in the trie and the strings that share a prefix intersect at some level of the trie. The leaf nodes contain the least significant bits of the bit strings (see Fig. 11).

Then, the algorithm determines the *maximal enclosing complete* subtries in the prefix trie (see Definition 2). A *maximal enclosing complete* subtrie essentially contains leaves (and its ancestors up to the root of the subtrie) corresponding to the integers that share an enclosing common prefix.

Definition 2. *A subtrie in a prefix trie is called a maximal enclosing complete subtrie if it satisfies the following properties: i) it is a complete binary tree, and ii) it is not part of any other complete binary subtree(s) rooted at one of its ancestors.*

Algorithm 1 Distance-Aware Set Augmentation

```
1:  Input
2:      $A$          Alice's input set
3:      $B$          Bob's input set
4:      $d_{\mathsf{int}}$      distance threshold
5:  Output
6:      Augmented sets $\hat{A}$, $\hat{B}$
7:  procedure GENERATE REPRESENTATIVE STRINGS FOR $A$
8:      for each $a_i \in A$ do
9:          for each integer, $a'_i \in (a_i - d, a_i + d)$ do
10:             $s'_i :=$ Bit string of $\mathsf{MaxBitLen}$ representing $a'_i$
11:         $P :=$ prefix trie with $s'_1, \ldots, s'_{2d_{\mathsf{int}}-1}$
12:         for each maximal enclosing complete subtrie $T$ in $P$ do
13:             $\phi :=$ prefix of $T$ in $P$
14:             Append "don't care" bits ($*$) to $\phi$ up to $\mathsf{MaxBitLen}$
15:             Add $\phi$ to $\hat{A}$
16:     return $\hat{A}$
17: procedure GENERATE REPRESENTATIVE STRINGS FOR $B$
18:     for each $b_j \in B$ do
19:         $s_j :=$ Bit string of $\mathsf{MaxBitLen}$ representing $b_j$
20: // Let $s_j = s_j[\mathsf{MaxBitLen}-1] \ldots s_j[0]$, $s_j[i]$ is the $i$th bit of $s_j$
21:         for $i = 0, 1, \ldots, \lfloor \log(2d-1) \rfloor$ do
22:             $s'_j := s_j[\mathsf{MaxBitLen}-1] \ldots s_j[i] * \ldots *$
23:             Add $s'_j$ to $\hat{B}$
24:     return $\hat{B}$
```

Each *maximal enclosing complete* subtrie corresponds to an enclosing common prefix of the bit strings for $(a - d_{\mathsf{int}}, a + d_{\mathsf{int}})$. Fig. 11 shows a prefix trie built over integers in the range [42,55]. Note that a leaf node in itself can be a *maximal enclosing complete* subtrie when the node is not part of any complete subtree. The algorithm identifies all the *maximal enclosing complete* subtries in the prefix trie corresponding to each $a_i \in A$. The prefix of each *maximal enclosing complete* subtrie is appended with wildcard bits up to the maximum bit length to form a representative string (Steps 12 - 15). The representative strings are added to the augmented set, $\hat{A}$.

For each $b_j \in B$, the algorithm generates representative strings by progressively replacing the least significant bits in the binary representation of b with wildcard bits. The process is repeated $\lfloor \log 2d - 1 \rfloor$ +1 times until a bit string is obtained by replacing the last $\lfloor \log 2d - 1 \rfloor$ +1 least significant bits . These strings are in the augmented set $\hat{B}$ (Steps 17 - 23).

Theorem 4. *There is a non-null intersection between the augmented sets obtained from Algorithm 1 if and only if there is some $(a_i, b_j) \in A \times B$ pair such that $|a_i - b_j| < d_{\mathsf{int}}$*

The proof is in the full version [3].

Number of Representative Strings: The following result shows that the number of representative strings for integers in the range $(a - d_{\mathsf{int}}, a + d_{\mathsf{int}})$ as a function of the distance parameter, d_{int} is $\mathrm{O}(\log d_{\mathsf{int}})$. This is analyzed by counting the number of *maximal enclosing complete* subtries in the prefix trie built over the binary representations of integers $(a - d_{\mathsf{int}}, a + d_{\mathsf{int}})$ since each such subtrie corresponds to an enclosing common prefix.

Theorem 5. *The total number of maximal enclosing complete subtries in a prefix trie built over the binary representations of all integers* $(a - d_{\mathsf{int}}, a + d_{\mathsf{int}})$ *is* $O(\log d_{\mathsf{int}})$.

Integer DA-PSI from PSI: Any existing private set intersection protocol can be used in conjunction with Algorithm 1 to provide a recipe for an integer distance-aware PSI protocol. In Section 6, we have instantiated an integer DA-PSI protocol with an OT-based PSI protocol due to Pinkas et al. [27]. We will omit details of this straightforward integration.

Theorem 6. *Assuming that there is a secure scheme for computing a private set intersection over sets of size n with* $O(n\lambda)$ *bits of communication. Then, there is a secure protocol realizing* $\mathcal{F}^{i\text{-}PSI}_{d_{\mathsf{int}}}$ *with* $O(n\lambda \log d_{\mathsf{int}})$ *bits of communication where* d_{int} *is the specified distance threshold.*

The proofs can be found in the full online version [3].

6 Evaluation

We have implemented both HamPSI (Sec. 4.2) and IntPSI (Sec. 5). In the following sections, we benchmark the protocols. The evaluation metrics are communication costs and compute times. All experiments consist of 5 independent trials and results are collected with a 95% confidence interval.

Platform: We ran our experiments on two different platforms representing low and high resource environments respectively.

- **Low-resource:** Unless stated otherwise, our experiments were run on two t2.xlarge Amazon EC2 instances with 4 vCPUs and 16GB of RAM. To simulate realistic scenarios, these instances were placed in different zones (US East and West). The network bandwidth between them was measured to be around 40–60MB per second using *iperf*[2].
- **High-resource:** We used a Microsoft Azure F72s_v2 instance, which has 72 virtual cores and 144GB of RAM, to compare to the results of Uzun et al. [32, Table 10].

6.1 Hamming Distance Protocol

Implementation: We have implemented HamPSI and HamPSISample in C++11. The implementation uses the NTL library[3] for implementing the finite field arithmetic, and the operations are performed over a 128-bit prime order field. We have used the open-source implementation[4] of the state of the art VOLE scheme [37] for our set reconciliation protocols.

Finally for tHamQueryRestricted, we have used the open source implementation[5] of the EC-ElGamal encryption scheme on a 256-bit curve as the additively homomorphic encryption. The scheme is set up with a 24-bit message space and a precomputed plaintext table of 24-bit messages to speedup the decryption process. The message space is large enough to encrypt the individual bits of the vectors, compute the Hamming distance between them over the ciphertexts and compare with the distance threshold. The 128-bit key returned in the protocol is split into 24-bit chunks to fit into the message space. More details are in App. A.1.

6.1.1 Comparison with Generic 2PC [15]

We have compared our Hamming DA-PSI protocol (Sec. 4.2) denoted HamPSI with the garbled circuit based construction by Huang et al. [14]. This construction is more efficient than the AHE-based scheme due to Osadchy et al. [25]. We use an open source implementation[6].

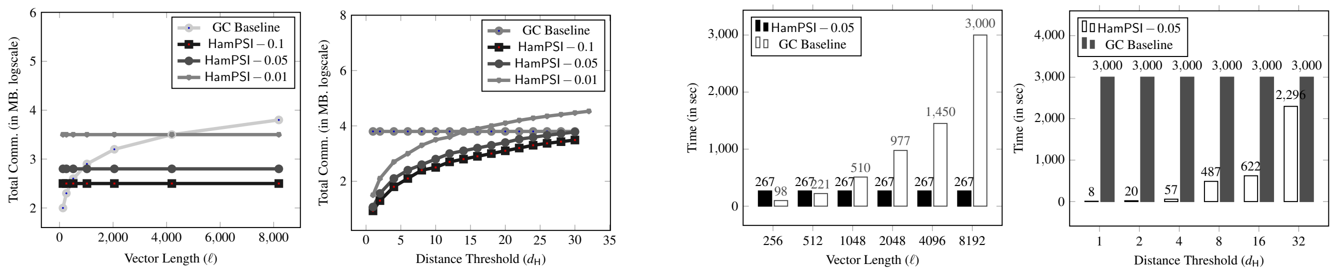
Micro-Benchmark: We run micro-benchmarks with sets containing 100 vectors sampled from the space $\{0,1\}^{\ell}$, and measure the communication volumes relative to the baseline.

- **Communication volume:** Fig. 12a shows how the communication volume scales with the the vector lengths, ℓ. The distance threshold $d_{\mathsf{H}} = 10$. As expected, the communication volume for HamPSI remains constant, while the communication volume for the GC based solution scales linearly in the vector size. For vectors of length greater than 512 bits, HamPSI outperforms the GC based solution. With 8192-bit vectors, HamPSI has 10× lower communication volume for FPR = 0.05. Fig. 12b shows how communication volume scales with the distance threshold, d_{H}. HamPSI has $1.5 - 440\times$ lower communication volumes up to $d_{\mathsf{H}} = 30$ compared to the baseline.
- **Compute time:** Fig. 12c and Fig. 12d shows how the compute time in HamPSI scales with vector lengths, and the distance threshold for FPR = 0.05. For vectors of length ≥ 1024 bits, HamPSI is at least 2× faster than the GC-based system. With 8192-bit vectors, HamPSI is faster than the GC-based system for distance threshold $d_{\mathsf{H}} \leq 32$.

Application Benchmark: We use Iris recognition as an application of HamPSI in a setting where the input vectors are long, while the distance threshold is small. In this setting, Alice and Bob have sets comprising 100 images of irises and want to learn if they have common elements. The iris data is collected from the CASIA dataset[7]. An open source tool is used to extract features from the dataset and compute 6000-bit long binary vectors corresponding to the items[8].

To privately compare a single pair of vectors, the garbled circuit based solution requires around 300KB of communication. With sets of size 100, the total communication required is around 3GB. The communication cost mainly depends on the vector length and is not affected by the distance threshold. For HamPSI, with a threshold $d_{\mathsf{H}} = 20$, we are able to retrieve all the matches with communication cost 2.5× and

[2] https://iperf.fr/
[3] https://libntl.org/
[4] https://github.com/emp-toolkit/emp-zk
[5] https://github.com/lubux/ecelgamal
[6] https://mightbeevil.org
[7] http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp
[8] https://github.com/mvjq/IrisRecognition



(a) Total comm. vs. vector length (b) Total comm. vs. threshold (c) Compute time vs. vector length (d) Compute time vs. threshold

Figure 12: Micro-benchmarks for HamPSI (Sec. 4.2). Set size $n = 100$. In Fig. 12a and Fig. 12b, the values on the y-axis are in logscale (base 10). Fig. 12a shows for $d_H = 10$, and for FPR = 0.05 and FPR = 0.01, HamPSI has around **10**× and **2**× lower communication volume respectively, compared to the GC baseline when the vector dimensions, $\ell = 8192$. Fig. 12b shows with $\ell = 8192$ dimension vector, and for FPR = 0.05, HamPSI has $\mathbf{2-537}\times$ lower communication volumes up to distance threshold $d_H = 32$ compared to the baseline. Fig. 12c shows that for vector lengths, $\ell \geq 1024$ bits, HamPSI is at least **2**× faster than the GC baseline. Fig. 12d shows that for up to distance threshold $d_H \leq 32$, HamPSI is faster than the GC baseline.

1.3× lower than the garbled circuit baseline with $\text{TNR} = 0.9$ and $\text{TNR} = 0.95$ respectively.

6.1.2 Comparison with Uzun et al. [32]

We have compared with the Hamming query protocol by Uzun et al. [32]. Their protocol has two components: an application-specific sub-sampling procedure that reduces bio-identifiers (e.g., bit vectors derived from facial features) to sets of high-dimensional items, and a t-out-of-T matching protocol. The purpose of our comparison is to show that our set reconciliation protocol is more efficient than the FHE-based t-out-of-T protocol of Uzun et al. [32]. In this way, the comparison is independent of the sub-sampling procedure, which can change based on the application.

Unfortunately, since we are unable to obtain their code[9], and compute results of their t-out-of-T matching protocol in isolation, for a fair comparison we use their sub-sampling procedure on top of OneSidedSetRecon and compare the overall times. This poses a challenge since the sub-sampling procedure outputs sets of sub-vectors, we cannot directly apply HamPSI which takes bit vectors as inputs. To overcome this problem we have compared their protocol with HamPSISample (Sec. 4.3, App. C) with their sub-sampling procedure replacing the mapping procedure. We stress that resorting to HamPSISample (vs. HamPSI) is simply to enable comparison to Uzun et al. [32] without their code. To implement a client-server containment query (as in [32]), Alice's input (client) is a singleton set $\{\vec{a}\}$ while Bob's input is $B := \{\vec{b}_1, \ldots, \vec{b}_n\}$. Alice's compute cost is $O\left(n \cdot \binom{T}{t}\right)$ where T, t are parameters of the sub-sampling procedure. When $T = 64$ and $t = 2$ [32], this cost is practically feasible.

Dataset: The dataset used for bechmarking by Uzun et al. [32] is a set of synthetically generated (by a generative network) images of human faces. Since biometric authentication/face recognition is not our focus, we opted to use a dataset with randomly generated bit vectors matching the parameters in [33]. Specifically, we generated sets of varying sizes containing bit-vectors of length $\ell = 256$, and then sub-sampled these bit vectors using the algorithm used by Uzun et al. [32] to generate corresponding sets of size $T = 64$.

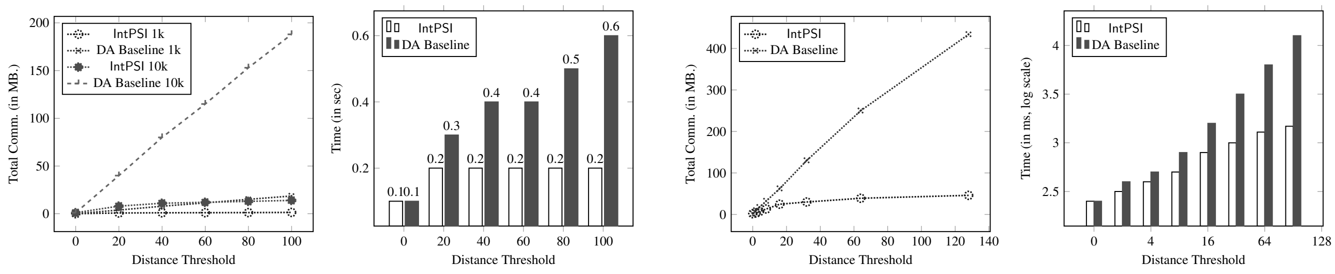
Results for High-Resource Setup: As recommended by the authors of [32], we have used our high resource setup when comparing our protocol with their protocol. Table 2 reports results of the comparison. The numbers for the protocol of Uzun et al. [32] correspond to a setup without load-balancing the dataset on the server, and so our results report the worst-case performance for both protocols. Load balancing can be applied to our protocol to improve performance; however, since it is unlikely to show improvements with randomly generated vectors, we omit this optimization.

Set size n	10K		100K		1M	
Measure	Comm	Comp	Comm	Comp	Comm	Comp
HamPSISample	25.8MB	2.00s	220MB	11.0s	1504MB	130s
Uzun et al. [32]	72.0MB	2.12s	528MB	17.8s	2124MB	189s

Table 2: Comparison of communication and computation costs of HamPSISample (App. C) with Uzun et al. [32].

Our protocol outperforms the protocol of Uzun et al. [32] for sets containing up to 1 million elements. The improvement is the result of using a communication-efficient VOLE protocol over fully homomorphic encryption. For instance, the amortized communication cost of performing a single oblivious linear evaluation as part of the VOLE protocol, i.e., computing a single value $\vec{z}[m] := \vec{u}[m]x + \vec{v}[m] \in \mathbb{F}_p$ (see the definition of $\mathcal{F}^p_{vole}$) for some $m \in [1, n]$, is roughly 3 bits [37]. When applied to the set containing 10K elements, we require 1.2 million correlations. The total communication cost is 3.6 million bits, or 450KB. In addition, we need to transfer a field

[9]The authors declined to provide the code for their implementation and instead recommended that we run our experiments on the same platform and compare our results to those reported in their paper.



(a) Comm vs. threshold. (b) Compute time vs. threshold (c) Comm vs. threshold (d) Compute time vs. threshold

Figure 13: Benchmarks for IntPSI. Fig. 13a and Fig. 13b are microbenchmarks with set sizes = 1k, 10k. The communication volume for IntPSI is roughly **13**× less than the baseline for threshold = 100. Compute time is **3**× less due to smaller augmented set sizes. Fig. 13c and Fig. 13d are benchmaks when IntPSI is applied to the task of private collaborative blacklisting. Set size = 25k. The communication volume and compute time for IntPSI are both around **10**× less than the baseline when threshold = 128.

element in plaintext to convert each pseudorandom VOLE correlation (as provided by Weng et al. [37]), to a correlation with the desired parameters. The total cost is 2.8× less than the cost of the protocol of Uzun et al. [32].

Compute costs are also lower due to the use of cheaper symmetric-key primitives in our protocol over fully homomorphic encryption. The only exception is for the set containing 10K elements since the VOLE protocol requires a setup time. For small sets, we still incur the setup time while not fully utilizing all the usable oblivious linear evaluations.

Set size n	10K		100K		1M	
Party	Alice	Bob	Alice	Bob	Alice	Bob
HamPSISample	11.4s	1.20s	24.3s	2.80s	145.3s	3.50s

Table 3: Compute costs of HamPSISample (App. C) on an Amazon t2.xlarge instance.

Results for Low-Resource Setup: The high-resource setup used by Uzun et al. [32] is necessary for the compute-intensive tasks in their FHE-based scheme. In fact, Table 8 of Uzun et al. [32] shows that deploying the system with 72 threads utilizing all the available vCPUs leads to a 32.4× speed up compared to a single-threaded deployment. Unlike their setting, PSI settings are often symmetrically provisioned and have more modest configurations. We demonstrate feasibility of our protocol even on low-resource platforms. And while we are unable to compute the actual costs of running the protocol of Uzun et al. [32] on the same platform, we posit that their compute costs would be significantly higher on low-resource systems due to the inherent cost of FHE.

Table 3 shows the compute times of our protocol on the low-resource platform described before. The communication costs remain the same as the ones presented in Table 2 and therefore we omit the results. Bob only participates in the VOLE protocol and therefore has no other compute costs. As is evident, the cost of our protocol on a low resource environment is similar to the performance on the over-provisioned system. The use of a cheap symmetric key primitive, namely VOLE, ensures that Bob's online compute times are low (less than 4s for databases containing up to 1M elements). The majority of Alice's time is spent on local computation which can be further optimized by leveraging parallel processing.

6.2 Integer Distance Protocol

Implementation: The integer distance-aware protocol (or IntPSI in short) is implemented as a two step process in C++. First both parties augment their sets using Algorithm 1. These augmented sets are used as inputs to the OT-based PSI protocol due to Pinkas et al. [27]. We rely on an open-source implementation[10]. *We note that* IntPSI *can be instantiated with any traditional PSI protocol of choice, and both communication volume and compute times are expected to show similar trends.*

Micro-benchmarks: We run micro-benchmarks with sets containing 1k and 10k elements each, randomly sampled from the space of non-negative (32 bit) integers. We evaluate how the communication and overall compute time of the protocol scales with the distance threshold and set size. The baseline is the protocol due to Pinkas et al. [27] where we augment the input sets with items in the neighborhood of each item in the set based on the threshold. Here, the augmented set size is expected to scale linearly with the distance threshold. This is called the "DA Baseline" in our experiments.

- **Comm. volume vs. threshold:** Fig. 13a shows how the communication volume scales with the distance threshold. The communication costs of IntPSI scale logarithmically, and so with distance threshold 100, the communication volume is 12.5× less than the baseline for set size 10K.
- **Compute time vs. threshold:** Fig. 13b shows how compute time scales with the distance threshold. Due to smaller set sizes to compute on, IntPSI compute time is 3× lower than the baseline when the threshold is 100.

Application Benchmark: To further explore realistic parameter settings, as a real-world application of IntPSI, we return

[10]https://github.com/encryptogroup/PSI

to the problem of private collaborative blacklisting where two mutually-untrusting parties compare IP addresses of endpoints from where they have observed traffic to their own network. This task is usually performed with a PSI protocol [22]. Replacing this with a DA-PSI protocol enables us to find IP addresses that are common to both sets, as well as find addresses that are "close" in the address space. A distance-based comparison is meaningful here because it is well-know that coordinated attacks usually span multiple subnets [5, 38].
Dataset: To test this application, we have collected data from a public honeypot deployed in a university network. The honeypot logs all incoming and outgoing traffic and stores a wealth of information. From this data, we curate information about traffic observed on two separate days, and build sets with the source IP addresses. There are roughly 25,000 distinct IP addresses in each set. These sets are inputs to IntPSI and the baseline PSI protocol.

We observe that using a distance-aware intersection in this context is well-justified based on the results. The number of intersections increases significantly when increasing the search radius and almost doubles by the time we reach the threshold of 128. In actual numbers, there are around 5% exact matches between the two sets. With a distance based search over thresholds of $2, \ldots 128$ we find that the number of items in the intersection increase to more than 10%. By searching over larger threshold, we are able to obtain matching IP addresses that fall in the same subnet/adjoining subnets. Note that without a full subnet map, it is not possible to predict these subnet sizes a priori. Therefore, we envision running the protocol multiple times with different thresholds to obtain the most informative intersection.
Results: Figs. 13c–13d show how the overall communication volume and compute time scale with distance thresholds set to $2, \ldots, 128$. The intuition behind increasing the threshold in powers of two is that we would like to search over entire subnets and find potential overlaps (if any). In terms of the overall runtime we observe that IntPSI scales more gracefully with the distance threshold. Note that with threshold set at 128, the baseline protocol computes over sets of size exceeding 6 million, while IntPSI computes over sets of size of around 200,000. This difference results in a significant speedup. With threshold = 128, IntPSI requires only 1.5 seconds to compute the intersection while the baseline requires over 15 seconds to accomplish the same task.

7 Conclusion

In this paper, we introduced the distance-aware PSI problem over metric spaces, whereby parties privately compute an intersection of their respective sets with items that are "close" in the metric space ending up in the intersection. Closeness is defined based on a user-specified distance threshold in the metric space. As concrete instantiations, we provided distance-aware constructions for two metric spaces: Minkowski distance of order 1 over the integers and Hamming distance. Both the protocols are communication-efficient. As a practical application of this idea, we evaluated the Minkowski distance protocol in the context of collaborative blacklisting. In addition, the Hamming distance-aware protocol allows constructions for other distances using techniques like locality-sensitive hashing.

Acknowledgments

This research was supported in part by grant numbers 2040675 and 1705007 from the National Science Foundation, and W911NF-17-1-0370 from the Army Research Office. This work was also made possible by the support of JP Morgan Chase, the Sloan Foundation, Siemens AG, and Cisco. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, Army Research Office, or the U.S. Government.

References

[1] S. Badrinarayanan, P. Miao, S. Raghuraman, and P. Rindal. Multi-party threshold private set intersection with sublinear communication. In *24th International Conference on Practice and Theory of Public Key Cryptography*, volume 12711 of *Lecture Notes in Computer Science*, pages 349–379, 2021.

[2] R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. Smith. Reusable fuzzy extractors for low-entropy distributions. *Journal of Cryptology*, 34(1):1–33, 2021.

[3] Anrin Chakraborti, Giulia Fanti, and Michael K. Reiter. Distance-aware private set intersection, 2021. URL https://arxiv.org/abs/2112.14737.

[4] M. Chase and P. Miao. Private set intersection in the internet setting from lightweight oblivious prf. In *Advances in Cryptology – CRYPTO 2020*, volume 12172 of *Lecture Notes in Computer Science*, pages 34–63, 2020.

[5] M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. De Shon, and J. Kadane. Using uncleanliness to predict future botnet addresses. In *7th ACM Conference on Internet Measurement*, 2007.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *20th Symposium on Computational Geometry*, 2004.

[7] John Daugman. Chapter 25 - how iris recognition works. In Al Bovik, editor, *The Essential Guide to Image Processing*. Academic Press, 2009.

[8] E. De Cristofaro, J. Kim, and G. Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, 2010.

[9] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *International conference on the theory and applications of cryptographic techniques*, pages 523–540. Springer, 2004.

[10] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19, 2004.

[11] S. Ghosh and M. Simkin. The communication complexity of threshold private set intersection. In *Advances in Cryptology – CRYPTO 2019*, volume 11693 of *Lecture Notes in Computer Science*, pages 3–29, 2019.

[12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *25th International Conference on Very Large Data Bases*, 1999.

[13] A. Groce, P. Rindal, and M. Rosulek. Cheaper private set intersection via differentially private leakage. *Proceedings on Privacy Enhancing Technologies*, 2019.

[14] Wei Huang, Yaoyun Shi, Shengyu Zhang, and Yufan Zhu. The communication complexity of the hamming distance problem. *Information Processing Letters*, 99 (4), 2006.

[15] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *20th USENIX Security Symposium*, 2011.

[16] B. Kacsmar, B. Khurram, N. Lukas, A. Norton, M. Shafieinejad, Z. Shang, Y. Baseri, M. Sepehri, S. Oya, and F. Kerschbaum. Differentially private two-party set operations. In *2020 IEEE European Symposium on Security and Privacy*, 2020.

[17] B. Kalyanasundaram and G. Schintger. The probabilistic communication complexity of set intersection. *Journal on Discrete Mathematics*, 5(4):545–557, 1992.

[18] L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, page 241–257, 2005.

[19] F. M. Larkin. Some techniques for rational interpolation. *The Computer Journal*, 10(2):178–187, 1967.

[20] G. S. Manku, A. Jain, and A. Das Sarma. Detecting near-duplicates for web crawling. In *16th International Conference on the World Wide Web*, 2007.

[21] C. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *IEEE Symposium on Security and Privacy*, 1986.

[22] L. Melis, A. Pyrgelis, and E. De Cristofaro. On collaborative predictive blacklisting. *ACM SIGCOMM Computer Communication Review*, 2019.

[23] Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9), 2003.

[24] M. Mohammadi-Kambs, K. Hölz, M. Somoza, and A. Ott. Hamming distance as a concept in dna molecular recognition. *ACS Omega*, 2(4):1302–1308, April 2017.

[25] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. Scifi - a system for secure face identification. In *31st IEEE Symposium on Security and Privacy*, 2010.

[26] B. Pinkas, T. Schneider, G. Segev, and M. Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium*, 2015.

[27] B. Pinkas, T. Schneider, and M. Zohner. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security*, 2018.

[28] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. Spotlight: Lightweight private set intersection from sparse ot extension. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 401–431, Cham, 2019. Springer International Publishing.

[29] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. Psi from paxos: Fast, malicious private set intersection. In *Advances in Cryptology -– EUROCRYPT 2020*, volume 12106 of *Lecture Notes in Computer Science*, pages 739–767, 2020.

[30] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *22nd International Conference on Neural Information Processing Systems*, 2009.

[31] J.F. Steffensen. Note on divided differences =. Mathematisk-fysiske Meddelelser. XVII, 3, 1939. `http://gymarkiv.sdu.dk/MFM/kdvs/mfm%2010-19/mfm-17-3.pdf`.

[32] E. Uzun, Simon P Chung, Vladimir Kolesnikov, Alexandra Boldyreva, and Wenke Lee. Fuzzy labeled private

set intersection with applications to private {Real-Time} biometric search. In 30th *USENIX Security Symposium*, .

[33] E. Uzun, C. Yagemann, S. Chung, V. Kolesnikov, and W. Lee. Cryptographic key derivation from biometric inferences for remote authentication. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, .

[34] K. C. Wang and M. K. Reiter. How to end password reuse on the web. In 26th *ISOC Network and Distributed System Security Symposium*, 2019.

[35] K. C. Wang and M. K. Reiter. Detecting stuffing of a user's credentials at her own accounts. In 29th *USENIX Security Symposium*, 2020.

[36] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu. Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In 22nd *ACM Conference on Computer and Communications Security*, 2015.

[37] C. Weng, K. Yang, J. Katz, and X. Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In 42nd *IEEE Symposium on Security and Privacy (SP)*, 2021.

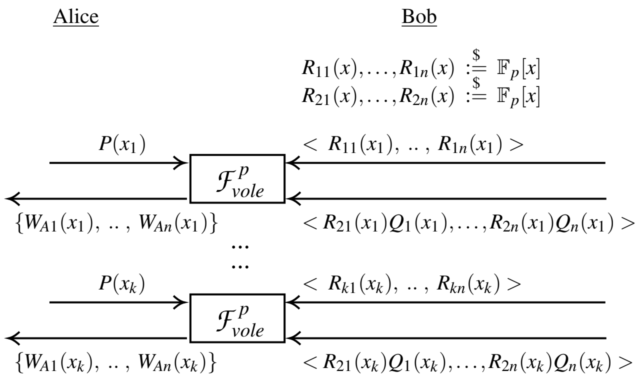
[38] A. G. West, A. J. Aviv, J. Chang, and I. Lee. Spam mitigation using spatio-temporal reputations from blacklist history. In 26th *Annual Computer Security Applications Conference*, 2010.

A tHamQuery: Threshold Hamming Queries (Sec. 4.1.3)

A.1 Using EC-Elgamal in tHamQuery

tHamQueryRestricted uses a additively homomorphic encryption scheme. In our implementation we use the EC-Elgamal encryption scheme to reduce communication costs. For this, the message space should be large enough to allow the computation in tHamQueryRestricted, and decrypt the results. We use a 24-bit message space. The computation in tHamQueryRestricted broadly involves two steps: i) computing the Hamming distance between two vectors as follows, and ii) returning a key κ blinded with the result of the Hamming distance computation. First, consider the mechanism we use to compute the Hamming distances between two vectors.

As long as the maximum Hamming distance between the vectors $< 2^{24}$, the 24-bit message space suffices for this computation. After computing $\mathsf{Enc}(\delta_H(\vec{a},\vec{b}))$, Bob computes $\mathsf{KeySet} := \{\kappa_i : \kappa_i := r_i \times (\mathsf{Enc}(\delta_H(\vec{a},\vec{b}) - i)) + \mathsf{Enc}(\kappa),\ i \in [0, d_\mathsf{H}],\ r_i \leftarrow \mathbb{F}_p\}$. However, the problem here is that $\kappa \in \mathbb{F}_p$ and for a sufficiently high statistical security parameter, we require p to be a at least 128-bit long. Thus, κ does not fit in the 24-bit message space.


Alice
Bob
$R_{11}(x),\ldots,R_{1n}(x) \overset{\$}{:=} \mathbb{F}_p[x]$
$R_{21}(x),\ldots,R_{2n}(x) \overset{\$}{:=} \mathbb{F}_p[x]$
$P(x_1)$
$< R_{11}(x_1),\,..\,,R_{1n}(x_1) >$
$\mathcal{F}^p_{vole}$
$\{W_{A1}(x_1),\,..\,,W_{An}(x_1)\}$
$< R_{21}(x_1)Q_1(x_1),\ldots,R_{2n}(x_1)Q_n(x_1) >$
...
...
$P(x_k)$
$< R_{k1}(x_k),\,..\,,R_{kn}(x_k) >$
$\mathcal{F}^p_{vole}$
$\{W_{A1}(x_k),\,..\,,W_{An}(x_k)\}$
$< R_{21}(x_k)Q_1(x_k),\ldots,R_{2n}(x_k)Q_n(x_k) >$
Figure 14: Using VOLE for OneSidedSetRecon

To mitigate this, we split κ into 24-bit chunks. Each individual chunk is encrypted separately, and returned to Alice. In other words, Bob splits κ into $c = \left\lceil \frac{|\mathbb{F}_p|}{24} \right\rceil$ chunks $\mathsf{Chunk}_1,\ldots,\mathsf{Chunk}_c$, and Alice now receives $\mathsf{KeySet} := \{\kappa_{ij} : \kappa_{ij} := r_{ij} \times (\mathsf{Enc}(\delta_H(\vec{a},\vec{b}) - i)) + \mathsf{Enc}(\mathsf{Chunk}_j),\ i \in [0, d_\mathsf{H}],\ r_{ij} \leftarrow \mathbb{F}_p, j \in [1,c]\}$. From this, Alice can obtain $\kappa_i := \kappa_{i1}||\ldots||\kappa_{ic}$. Note each 24-bit chunk in encrypted with IND-CPA security, and therefore splitting the key as described has no impact on security. There is a c times blowup in the downstream communication cost i.e., the cost of sending KeySet to Alice.

B HamPSI: Hamming DA-PSI from tHamQuery (Sec. 4.2)

For inputs, $A := A := \{\vec{a}_1,\ldots,\vec{a}_n\}$ and $B := B := \{\vec{b}_1,\ldots,\vec{b}_n\}$, the straightforward way to realize a Hamming DA-PSI protocol is by running n^2 instances on tHamQuery in parallel over each input pair $(\vec{a}_i,\vec{b}_j)$. However, there is more optimized solution using vector OLE's (see Sec. 3.1). The idea is as follows: consider Alice's input is $\vec{a}_i$ and Bob's input is the set of vectors $B := \{\vec{b}_1,\ldots,\vec{b}_n\}$. Then, the goal is to determine if there exists $\vec{b}_j \in B$ such that $\delta_H(\vec{a}_i,\vec{b}_j) < d_\mathsf{H}$. This functionality can be considered a *one-sided containment query*.

To realize this functionality, we present HamContainQuery protocol (see Fig. 15), consisting of two procedures PermuteAndPartition and OneSidedSetReconMultiBlind. In PermuteAndPartition, Alice and Bob run tHamQueryRestricted over each pair $(\vec{a}_i,\vec{b}_j)$ (lines 1–2). At the end of this procedure, Alice obtains $\{\mathsf{KeySet}_1,\ldots,\mathsf{KeySet}_n\}$. Alice obtains the set of sub-vectors $S'_{\vec{a}} := \{\vec{x}_1,\ldots,\vec{x}_{N_\mathsf{bins}}\}$ derived from $\vec{a}$. Similarly, for $i \in [1,n]$, Bob obtains the set $S'_{\vec{b}_i}$ derived from $\vec{b}_i$ (line 3).

OneSidedSetReconMultiBlind takes as input S_A from

Alice while Bob's input is a set of sets $\{S_{B1},\dots,S_{Bn}\}$. Alice derives $P(x)$ from S_A and Bob derives the set of polynomials $\{Q_1(x),\dots,Q_n(x)\}$from $\{S_{B1},\dots,S_{Bn}\}$ (line 4). Bob selects two sets of degree-ℓ random polynomials $\{R_{11}(x),\dots,R_{1n}(x)\}$, $\{R_{21}(x),\dots,R_{2n}(x)\}$. Then, for $k \in [1, N_{\mathsf{bins}}+2d_{\mathsf{H}}+1]$, Alice and Bob compute $\{W_{A1}(x_k),\dots,W_{An}(x_k)\}$ where $W_{Ai}(x_k) := P(x_k)R_{1i}(x_k) + R_{2i}(x_k)Q_i(x_k) + \phi(\kappa_i,k)$.

In order to compute $\{W_{A1}(x_k),\dots,W_{An}(x_k)\}$, the idea is to use a VOLE protocol (see Fig. 14). Specifically, for $k \in [1, \ell+2d_{\mathsf{H}}+1]$ Alice sends $P(x_k)$ to $\mathcal{F}^p_{vole}$, while Bob sends the two vectors $\vec{R} :=< R_{11}(x_k),\dots,R_{1n}(x_k) >$, and $\vec{U} :=< R_{21}(x_k)Q_1(x_k) + \phi(\kappa_1,k),\dots,R_{2n}(x_k)Q_n(x_k) + \phi(\kappa_n,k) >$ (line 6). By definition, $\mathcal{F}^p_{vole}$ return to Alice $\{W_{A1}(x_k),\dots,W_{An}(x_k)\}$. Batching n OLE computations with a single VOLE instance has concrete advantages both in terms of compute time and communication costs. Fig. 14 describes this process. Subsequently, for $\kappa' \in \mathsf{KeySet}_i$ and $i \in [1,n]$, Alice computes $V_{i\kappa'} = \{(x_k,y_k) : y_k = \frac{W_{Ai}(x_k)-\phi(\kappa',k)}{P(x_k)},\ k \in [1, N_{\mathsf{bins}}+2d_{\mathsf{H}}+1]\}$, and interpolates the set of points (line 8).

Finally, realizing a Hamming PSI protocol is straightforward with a one-sided containment query protocol. Specifically, we run n instances of containment queries corresponding to each element in A. Each instance is run independently with independent random coins (see Fig. 16).

Theorem 3. *Assuming that there exists a semantically secure additively homomorphic encryption scheme, and a protocol securely realizing $\mathcal{F}^p_{vole}$ with $O(n\lambda)$ bits of communication, there is a Hamming DA-PSI protocol which securely realizes $\mathcal{F}^{h\text{-}PSI}_{\ell,d_{\mathsf{H}}}$ with $O\left(n^2 \cdot \frac{d^2_{\mathsf{H}}}{\mathrm{FPR}} \cdot \lambda\right)$ bits of communication where* $\mathrm{FPR} \in (0, 0.5)$ *is the false positive rate.*

The proof can be found in the full version [3].

C HamPSISample: Hamming DA-PSI with Sub-Sampling (Sec. 4.3)

This section details a construction for Hamming queries which combines OneSidedSetReconExp with a sub-sampling algorithm which makes the computation feasible.

Sub-Sampling: There is extensive work on reducing bit vectors to sets of small sizes for Hamming distance comparisons in specific application settings e.g., in biometric authentication [32, 33]. The idea is to sub-sample the bit vectors of length ℓ into T sub-vectors, and then compare the sets of the sub-vectors. If the input bit vectors are close in Hamming space, then t out of the T sub-vectors will match across the sets. The sub-sampling scheme is parameterized such that $t \ll T < \ell$. For example, in the context of biometric data represented by bit vectors of length $\ell = 256$ after a transformation with a locality-sensitive hash, the sub-sampling algorithm can yield sets of size $T = 64$ as inputs, with $t = 2$.

<u>**Parameters:**</u> Alice holds vector $\vec{a}$ and Bob has set of vectors $B := \{\vec{b}_1,\dots,\vec{b}_n\}$. They jointly select a Hamming distance threshold $d_{\mathsf{H}} \in (0, \ell/2)$.

<u>**Procedure** PermuteAndPartition**:**</u>

(1) Bob samples n random keys $\{\kappa_1,\dots,\kappa_n\}$ where $\kappa_i \overset{\$}{:=} \mathbb{F}_p$.

(2) For each $i \in [1,n]$, Alice sends $\vec{a}$ and Bob sends $\vec{b}_i$ and κ_i to tHamQueryRestricted. Alice obtains $\{\mathsf{KeySet}_1,\dots,\mathsf{KeySet}_n\}$.

(3) Let $N_{\mathsf{bins}} := \frac{2d^2_{\mathsf{H}}}{\mathrm{FPR}}$. Alice computes set $S'_{\vec{a}} := \{\vec{x}_1,\dots,\vec{x}_{N_{\mathsf{bins}}}\}$ using π similar to line 3 of tHamQueryRestricted. . Similarly, for each $i \in [1,n]$, Bob computes set $S'_{\vec{b}_i}$.

<u>**Procedure** OneSidedSetReconMultiBlind**:**</u>

(4) Alice computes the polynomial $P(x) := \prod_{r \in S_{\vec{a}}} (x-r)$. For each $i \in [1,n]$, Bob computes set $Q_i(x) := \prod_{r \in S'_{\vec{b}_i}} (x-r)$.

(5) Bob samples two sets of n polynomials $\{R_{11}(x),\dots,R_{1n}(x)\}$, and $\{R_{21}(x),\dots,R_{2n}(x)\}$ where each $R_{1i}(x)$, $R_{2i}(x) \overset{\$}{:=} \mathbb{F}_p[x]$ are degree-(N_{bins}) polynomials.

(6) For each $k \in [1, N_{\mathsf{bins}}+2d_{\mathsf{H}}+1]$, Alice and Bob engage in a round $\mathcal{F}^p_{vole}$. Specifically, Alice sends $P(x_k)$ to $\mathcal{F}^p_{vole}$ and Bob sends the vectors $< R_{11}(x_k),\dots,R_{1n}(x_k) >$, and $< R_{21}(x_k)Q_1(x_k) + \phi(\kappa_1,k),\dots,R_{2n}(x_k)Q_n(x_k) + +\phi(\kappa_n,k) >$. Alice obtains $\{W_{A1}(x_k),\dots,W_{An}(x_k)\}$ where $W_{Ai}(x_k) := R_{1i}(x_k)P(x_k) + R_{2i}(x_k)Q_i(x_k) + \phi(\kappa_i,k)$.

(7) For each $\kappa' \in \mathsf{KeySet}_i$, Alice computes the set of sets $\{V_1,\dots,V_n\}$, where
$$V_{i\kappa'} = \{(x_k,y_k) : y_k = \frac{W_{Ai}(x_k)-\phi(\kappa',k)}{P(x_k)}\}.$$

(8) Alice runs the interpolation in line 6 in Fig. 3 over $V_{i\kappa'}$ and returns $\{i,\kappa'\}$ if the interpolation succeeds. Otherwise, Alice returns $\perp$.

Figure 15: HamContainQuery: Hamming containment query protocol

In the typical setting where Alice and Bob hold bit vectors, $\vec{a}$ and $\vec{b}$ of length ℓ, Bob samples T "masking" functions, $\{mask_1,\dots,mask_T\}$. These masking functions are applied to $\vec{a}$ using a 2PC circuit (e.g., with a garbled circuit or an OPRF) to create a set of sub-vectors $S_{\vec{a}} := \{PRF_{\kappa_s}(\vec{a} \bigwedge mask_1),\dots,PRF_{\kappa_s}(\vec{a} \bigwedge mask_T)\}$. Here, PRF is a keyed PRF e.g., AES with κ_s as the key, uniformly sampled by Bob. In this way, Alice does not learn the masking functions but learns $S_{\vec{a}}$ as output of the 2PC circuit. Bob similarly applies the masking functions to his own inputs, $S_{\vec{b}} := \{PRF_{\kappa_s}(\vec{b} \bigwedge mask_1),\dots,PRF_{\kappa_s}(\vec{b} \bigwedge mask_T)\}$. Next, Alice and Bob run a threshold PSI (t-out-of-T matching) algorithm over $S_{\vec{a}}$ and $S_{\vec{b}}$. If t elements match in $S_{\vec{a}}$ and $S_{\vec{b}}$, Alice learns that $\vec{a}$ and $\vec{b}$ are close in context of the application. We refer to existing work [2, 9, 32, 33] for further details, and focus on the threshold PSI part of this process. Fig. 17 describes the protocol. After creating the sets by sub-sampling the input vectors in line 2 of Fig. 17, Alice and Bob run OneSidedSetReconExp with $\ell = T, d_{\mathsf{H}} = (T-t)$.

Optimizing the Interpolation: In line 10 of Fig. 17, the interpolation step can be optimized based on two insights. The first insight is based on the fact that we are only interested in the degree of the interpolating polynomial *and not the polyno-*

Inputs: Alice holds a set of vectors $A := \{\vec{a}_1, \ldots, \vec{a}_n\}$. and Bob has set of vectors $B := \{\vec{b}_1, \ldots, \vec{b}_n\}$. They jointly select a Hamming distance threshold d_{H}.
Protocol:
(1) For each $i \in [1,n]$, Alice and Bob run HamContainQuery with inputs $\vec{a}_i$ and $B := \{\vec{b}_1, \ldots, \vec{b}_n\}$, and distance threshold d_{H}.
(2) For $i \in [1,n]$, if HamContainQuery returns $\{j, \kappa'\}$ for input $\vec{a}_i$, Alice sends to Bob $\{\vec{a}_i, j, \kappa'\}$. Bob verifies i) κ' is one of the chosen keys in line 1, and ii) $\delta_H(\vec{a}_i, \vec{b}_j) < d_{\mathsf{H}}$. If so, Bob returns $\vec{b}_j$. Alice add $(\vec{a}_i, \vec{b}_j)$ to the set S_{out}.
(3) Alice outputs S_{out}.

Figure 16: HamPSI: Hamming distance-aware PSI protocol

<u>**Parameters:**</u> Alice and Bob have vectors $\vec{a}, \vec{b} \in \{0,1\}^{\ell}$ respectively and a hamming distance threshold d_{H}.
<u>**Procedure** SubSample:</u>
(1) Bob samples T random masking functions $\{mask_1, \ldots, mask_T\}$ [32], and keyed PRF, PRF_{κ_s} e.g., AES with key $\kappa_s := \{0,1\}^{\lambda}$.
(2) Alice and Bob run a 2PC circuit where Alice's input is $\vec{a}$ and Bob's input is $\{mask_1, \ldots, mask_T\}$, and PRF_{κ_s}. The circuit returns to Alice $S_{\vec{a}} := \{PRF_{\kappa_s}(\vec{a} \wedge mask_1), \ldots, PRF_{\kappa_s}(\vec{a} \wedge mask_T)\}$.
(3) Bob locally computes $S_{\vec{b}} := \{PRF_{\kappa_s}(\vec{b} \wedge mask_1), \ldots, PRF_{\kappa_s}(\vec{b} \wedge mask_T)\}$.
<u>**Procedure** OneSidedSetReconExp:</u> Alice and Bob run OneSidedSetReconExp with $S_{\vec{a}}$ and $S_{\vec{b}}$ as inputs.
(4) Alice and Bob select set of points $X = \{x_k\}_{k=1}^{2T-t+2}$.
(5) Alice encodes $S_{\vec{a}}$ in roots of the polynomial $P(x) = \prod_{r \in S_{\vec{a}}} (x-r)$ and Bob encodes $S_{\vec{b}}$ in roots of the polynomial $Q(x) = \prod_{r \in S_{\vec{b}}} (x-r)$.
(6) Bob samples two random polynomials $R_1(x), R_2(x) \overset{\$}{:=} \mathbb{F}_p[x]$ of degree T.
(7) For $k \in [1, 2T-t+2]$, Alice sends $P(x_k)$ to $\mathcal{F}_{ole}^{p}$ and Bob sends $R_1(x_k)$ and $R_2(x_k)$ $Q(x_k)$. Alice learns $W_A(x_k) := R_1(x_k)P(x_k) + R_2(x_k)Q(x_k)$ as the output of $\mathcal{F}_{ole}^{p}$.
(8) Alice computes set C_{sub} where each $C_i \in C_{\mathsf{sub}}$ is a subset of $S_{\vec{a}}$ having exactly t elements.
(9) For each $C_i \in C_{\mathsf{sub}}$, Alice computes the polynomial $P_i := \prod_{r \in C_i} (x - r)$ and computes the set of points
$$V_i = \{(x_k, y_k) : y_k = \tfrac{W_A(x_k)}{P_i(x_k)},\ k \in [1, 2T-t+2]\}.$$
(10) Alice interpolates V_i for all $i \in |C_{\mathsf{sub}}|$ with a polynomial. If the degree of the interpolating polynomial is $< 2T - t$, Alice learns that $\vec{a}$ and $\vec{b}$ are close. Otherwise, Alice outputs $\perp$.
<u>**Procedure** Recover:</u>
If OneSidedSetReconExp outputs $|S_{\vec{a}} \setminus S_{\vec{b}}|$ then obtain $\vec{b}$ from Bob. Output $(\vec{a}, \vec{b})$. Otherwise, output $\perp$.

Figure 17: tHamQuerySample: Threshold Hamming queries with sampling

mial itself. Therefore, instead of computing the interpolating polynomial completely we can compute the coefficients of the constituent monomials of Newton's interpolating polynomial. More specifically, the interpolating polynomial is of the form

$$R(x) := f[x_1] + f[x_1,x_2](x-x_1) + \ldots + f[x_1,\ldots,x_{2T-t+2}](x-x_1)(x-x_2)\ldots(x-x_{2T-t+2})$$

Here, $f[x_1], f[x_1,x_2]\ldots$ are the Newton's divided differ-

Inputs: Alice holds a set of vectors $A := \{\vec{a}_1, \ldots, \vec{a}_n\}$. and Bob has set of vectors $B := \{\vec{b}_1, \ldots, \vec{b}_n\}$. They jointly select a Hamming distance threshold d_{H}.
Protocol:
(1) For each $i \in [1,n], j \in [1,n]$, Alice and Bob run tHamQuerySample with inputs $\vec{a}_i$ and $\vec{b}_j$, and distance threshold d_{H}.
(2) For $i \in [1,n]$, $j \in [1,n]$, if tHamQuerySample returns $(\vec{a}_i, \vec{b}_j)$, then add it S_{out}.
(3) Alice outputs S_{out}.

Figure 18: HamPSISample: Hamming distance-aware PSI protocol from tHamQuerySample

ences computed from the points in V_i. Since, we only need the coefficient of x^{2T-t+1} in this polynomial, it suffices to compute the value of $f[x_1,\ldots,x_{2T-t+2}]$. If $f[x_1,\ldots,x_{2T-t+2}] = 0$, the degree of $R(x)$ is $< 2T-t+1$.

Consider the divided differences:

(1) $f'[x_1], f'[x_1,x_2], \ldots, f'[x_1,\ldots,x_{2T-t+2}]$, over the points $\{(x_k,y_k) : y_k := W_A(x_k),\ k \in [1, 2T-t+2]\}$

(2) $g[x_1], g[x_1,x_2], \ldots, g[x_1,\ldots,x_{2T-t+2}]$ over the points $\{(x_k,y_k) : y_k := \frac{1}{P_i(x_k)},\ k \in [1, 2T-t+2]\}$

Since $R(x) = W_A(x) \times \frac{1}{P_i(x)}$, the divided differences for $R(x)$ can be computed using the divided differences for $W_A(x)$ and $\frac{1}{P_i(x)}$ due to Leibniz's rule [31]. That is,

$$f[x_1,\ldots,x_{2T-t+2}] = (f'.g)[x_1,\ldots,x_{2T-t+2}] = \sum_{k=1}^{2T-t+2} f'[x_1,\ldots,x_k]g[x_k,\ldots,x_{2T-t+2}] \quad (1)$$

The second insight is based on the fact that the divided differences for all the polynomials $P_i(x)$'s tested in lines 9–10 can be pre-computed offline by Alice and stored for speeding up the computation in (1). Specifically, Alice computes for each polynomial $P_i(x)$ (generated from $C_i \in C_{\mathsf{sub}}$, the set comprising the divided differences $\{g[x_1,\ldots,x_{2T-t+2}], g[x_2,\ldots,x_{2T-t+2}], \ldots, g[x_{2T-t+2}]\}$. This is done offline. Subsequently, when Alice obtains the points for $W_A(x)$ after executing line 7, she computes the divided differences for $W_A(x) \times \frac{1}{P_i(x)}$ using (1). This significantly speeds up the compute times.

Theorem 7. *Assuming that there is a protocol securely realizing $\mathcal{F}_{ole}^{p}$ with communication cost scaling linearly with the vector size and a sub-sampling algorithm which derives sets of size T after sub-sampling binary vectors of length ℓ such that sets corresponding to the vectors close in Hamming space have at least t common elements, tHamQuerySample securely realizes $\mathcal{F}_{\ell,d_{\mathsf{H}}}^{t\text{-}HQ}$ with $\mathrm{O}(T)$ communication cost and $\mathrm{O}\left(\binom{T}{t}\right)$ compute costs.*

The proof can be found in the full version [3].