Privately Matching k-mers

Justin Bedő[1,3], Thomas Conway[2], Kim Ramchen[1], and Vanessa Teague[1]

[1]Department of Computing and Information Systems, the University of Melbourne, {bedoj, kim.ramchen, vjteague}@unimelb.edu.au
[2]IBM Research Australia, tconway@au1.ibm.com
[3]The Walter and Eliza Hall Institute

Abstract

We construct the first noninteractive protocols for several tasks related to private set intersection. We provide efficient protocols for three related problems, each motivated by a particular kind of genomic testing. *Set intersection with labelling* hides the intersecting set itself and returns only the labels of the common elements, thus allowing a genomics company to return diagnoses without exposing the IP of its database. *Fuzzy matching with labelling* extends this to allow matching at a particular Hamming distance, which solves the same problem but incorporates the possibility of genetic variation. *Closest matching* returns the item in the server's database closest to the client's query - this can be used for taxonomic classification. Our protocols are optimised for the matching of k-mers (sets of k-length strings) rather than individual nucleotides, which is particularly useful for representing the short reads produced by next generation sequencing technologies.

1 Introduction

While personal genome sequencing projects have opened up many exciting possibilities, significant challenges are posed in reconciling the conflicting goals of broad accessibility to diverse genomic data sets and the need for privacy.

Malin and Sweeney [1] have showed that phenotypic information (gender, hair colour *etc.*) may be extracted from DNA databases such as those publically available on the internet or those compiled for clinical and research purposes, while Gymrek *et al.* [2] demonstrated the possibility of recovering surnames by profiling short tandem repeats (STRs) on the Y chromosome and querying recreational genealogy databases. Furthermore Homer *et al.* [3] show how to determine an individual's presence in a case or control group from aggregate allele frequencies such as those found in Genome Wide Association Studies (GWAS). Even publishing aggregate statistics such as p-values and coefficients of determination can serve to identify DNA markers unique to an individual [4]. Evidencing the second problem, genomic information via a phenomenon called linkage disequilibrium has been used to infer SNPs[1] of relatives [5] as well as predict genetic risk for disease even when the genomic region surrounding the associated transcript has been redacted [6].

1.1 Privately Mining Genomes

Previous works on homomorphic computation for genomic computations have typically focused on the representation of genotype as a list of deltas from a reference genome, such as one found in a variation call format (VCF) file. In this work we use raw k-mers of nucleotides, *i.e.*, all DNA subsequences of length k from a DNA sequence. This is particularly useful for representing short reads produced by next generation sequencing technologies and can capture more complex structural variations than SNPs alone [7].

Short DNA sequences (i.e. k-mers) can be used as markers for disease in patients, or as identifying markers for pathogens. For example, certain mutations are associated with particular forms of epilepsy [8], and we can differentiate strains of a pathogen such as *chlamydia trachomatis* [9]. In both cases, but especially in the latter case of bacterial pathogens, natural variations in the DNA sequences mean that we may wish to look for DNA sequences that are similar but not necessarily identical to a set of reference marker sequences.

The client is a patient who wishes to receive a diagnosis based upon a set of raw k-mers of their DNA provided by a genomic sequencing service. The server is a test provider which holds biomarker sequences with attached label information. The patient wishes to learn the result of the test (a set of labels) without disclosing either their genome or the test result, while the provider does not wish to disclose which biomarker sequences are associated with which label.

The client generates a public key under a homomorphic encryption scheme and then publishes *online* encryptions of all k-mers under this key to a diagnostic service (provider), for example a pathology lab or hospital. The provider having received all necessary input is then able to compute homomorphically *offline* and return the result, still in encrypted form, to the patient. The patient uses their secret key to decrypt the ciphertext, yielding the diagnosis. The provider, on the other hand, learns nothing about either the diagnosis nor the input used to determine it.

1.2 Our Contributions

We consider several tasks related to private matching of k-mers:

- **Set Intersection with labelling.** A patient (client) holds a set of k-mers X, while the provider (server) holds a set of k-mers Y. Associated with each k-mer in the provider set is a label holding diagnostic information. The patient should learn the labels associated with the k-mers in the intersection of the two sets.

[1] Single Nucleotide Polymorphisms

- **Fuzzy Matching with labelling.** The client holds a set of k-mers X, while the provider holds a set of k-mers Y. There is a label associated with each k-mer in the provider set. The patient should learn the labels associated with the k-mers in the provider set which are within a target set of Hamming distances from those k-mers in the patient set.

- **Closest Matching.** The client holds a k-mer x; the server holds a set of k-mers Y. The client should learn the k-mer in Y with minimum Hamming distance from x.

We construct simple low-depth circuits for these tasks based upon adaptation of protocols for private set intersection by Freedman *et al.* [10] and Kissner and Song [11] respectively. Using the YASHE levelled homomorphic encryption scheme [12] allows an efficient *noninteractive* solution: the client simply sends an appropriately encrypted representation of the genome, the server computes on it homomorphically and returns an encrypted result. We have implemented each protocol using the SEAL library [13] and found that they run efficiently enough for practical use, taking a few minutes of client computation and a few hours on the server side. Our set intersection with labelling is noticeably more efficient than the prior best [14].

(A *levelled homomorphic encryption scheme* [12, 15–19] permits a pre-specified number of levels of computation. In practice LHE schemes can be much more efficient than fully homomorphic encryption.)

In this setting this is a substantial improvement over prior set intersection protocols [20,21] based on Oblivious Transfer [22], which require multiple rounds of interaction. Our best protocol evaluates a circuit of depth $O(\log |Y|)$ and requires only a linear number of operations by client and server, which is significantly more efficient than both Freedman *et al.* and Kissner and Song which are based upon additively homomorphic encrypion. Our protocol for fuzzy matching is the only noninteractive protocol we know of, and the only one to allow efficient matching of a set of Hamming distances that need not necessarily be contiguous.

Another trend in recent years has been the construction of efficient maps supporting batch encryption of plaintext data with corresponding homomorphic single instruction multiple data (SIMD) operations. The availability of simple generic SIMD routines for computing on encrypted data promises to make homomorphic encryption an integral part of privacy preserving computing, much like the existence of such routines in hardware for parallel computing has already become the mainstay of computer graphics and high performance scientific computing communities.

Biological motivation

The first task is motivated by the scenario where the biomarkers are derived from an organism with a highly conserved genome. In this case there is little variation between sequences derived from any two individuals, hence exact matching between a patient's genome and a library of biomarkers is informative and dependable.

When considering organisms with a higher degree of variation (e.g., bacterial pathogens), the exact matching approach is no-longer viable: some degree of fuzzy matching is required to allow biomarker sequences to still match under less genomic conservation. Such an approach is suitable for tests such as serovar identification in pathogens such as *Chlamydia trachomatis*, an important piece of information for clinical decision making. Our second task is motivated by this and we describe a novel circuit computing the underlying functionality based upon an equivalence between Hamming distance and the component-wise sum over the integers of the cyclic shifts of the xor-sum of the inputs. This circuit is low-depth (depth two in fact), and is amenable to homomorphic SIMD operations, requiring only $O(k)$ SIMD sums, differences and multiplications.

Finally, the last task is motivated by taxonomic classification. Here, the closest match to a query sequence is sought as the identity of the closest match may reveal important taxonomic information for tasks such as species identification. For this task, we describe an efficient circuit based upon the addition of Hamming weights by permutation matrices and a binary tree search with a custom comparison operation. All of the circuits above are, via homomorphic encryption, ported to the two-party private function evaluation context. The evaluation of the corresponding circuits yields simple and efficient protocols for these functionalities which, with suitable de-randomization techniques [10] can be made secure against malicious adversaries. For the final task, closest matching, we describe a pre-computation approach to reducing the total noise in the homomorphic evaluation which may be of interest elsewhere.

The use of an arbitrary labelling function for the first two tasks allows significant flexibility in accommodating distinct but related tasks. For example, substituting diagnostic labels with actual k-mers, yields protocols for private set intersection and private fuzzy matching respectively.

Our main contribution is in the construction of secure protocols realising each of these tasks, we do not claim any improvements upon the YASHE LHE scheme [12].

1.3 Related Work

Prior work on Private Genomic Computation

Kim *et al.* [23] showed how to privately compute minor allele frequencies and a chi-squared test on case and control groups in Genome-Wide Associate Studies. Additionally they showed how to privately compute the Hamming distance and approximate edit distance between two encrypted genome sequences encoded from VCF files. Cheon *et al.* [24] showed how to compute exact edit distance on homomorphically encrypted data. Yasuda *et al.* [25] described a packing method for efficiently computing multiple Hamming distance values on encrypted data. Ayday *et al.* [26] assessed genetic risk based upon a sum of SNPs weighted by importance, their total model also assumes the availability of external clinical and environmental data. Danezis *et al.* [27] describe a scalable extension to this protocol via a third-party assisted storage-and-processing unit secured with a smart-card belonging to patient being tested. Neither protocol makes any attempt to hide the set of candidate SNPs used in their risk estimations. De Cristofaro *et al.* [14] describe size and position hiding sequence matching against a target genome, their approach based on additively homomorphic encryption requires computing a product on ciphertexts for every substring of the target.

Bruekers *et al.* [28] propose private identity and ancestry testing via private matching of STR profiles as tuples of multi-sets, pre-computation allows their protocols to run in a constant number of rounds.

A fundamental limitation of the above protocols is that either they fail to accommodate approximate matching according to a threshold distance (detailed in Section 3 following) at all, or their solutions would suffer an exponential blow-up in computation/communication costs in this parameter, rendering them unusable in practice. An additional concern is that no prior considered protocols for private DNA substring matching with remote diagnosis in mind support the private transference of associated descriptive labels, thus limiting their accessibility to the end user. We note that cryptographic protocols with different applications in mind exist that could conceivably be adapted to solve the tasks we consider in Section 3. For example, protocols exist for computing private set intersection [20, 21] which are based upon variants of oblivious transfer, and protocols computing private fuzzy matching [29, 30], which are based upon linear secret-sharing. On the other hand these protocols require multiple rounds of interaction and are thus unsuited to our offline/online problem setting. This motivates the need for efficient, scalable *non-interactive* solutions based solely on public-key primitives.

Prior work on Homomorphic Evaluation of Circuits

Gentry *et al.* [16] were the first to apply homomorphic encryption to evaluating a block cipher. In their original implementation, they showed that the full AES circuit could be evaluated in a practical amount of time, while subsequent optimizations have yielded substantially better results. Since then several works have shown how to evaluate AES and other block ciphers in other levelled homomorphic encryption schemes. Lepoint *et al.* [31] show to evaluate SIMON-64/128 using the Fan-Vercauteren [19] and YASHE [12] cryptosystems, as well as how to choose securely choose parameters for their implementations. Alperin-Sheriff *et al.* [32] show how to circumvent evaluation of the boot-strapping circuit as a long branching program, by replacing decryption with a simple arithmetic circuit over a large modulus q. A key feature of their construction is the computation of addition over the cyclic group $\mathbb{Z}_q$ as multiplication of $q \times q$ permutation matrices. This idea is a core concept in the efficient homomorphic computation and comparison of Hamming weights in our third protocol.

2 Preliminaries

We first survey the necessary background to work with the levelled homomorphic scheme YASHE [12] as well as the coding theory background required to understand our second and third protocols on fuzzy matching with labelling and minimum matching.

Let R be the cyclotomic ring $\mathbb{Z}[X]/(\Phi_d(x))$ for d a positive integer. The degree of Φ_d is $n = \phi(d)$, where ϕ is Euler's totient function. The elements of R can be uniquely represented by all polynomials in $\mathbb{Z}[X]$ of degree at most $n-1$. An arbitrary element $a \in R$ can be written as $a = \sum_{i=0}^{n-1} a_i X^i$ with $a_i \in \mathbb{Z}$ and we identify a with its vector of coefficients

$(a_0, a_1, \ldots, a_{n-1})$. Thus a can be viewed as an element of real vector space $\mathbb{R}^n$. The maximum norm on $\mathbb{R}^n$ is used to the measure the size of elements in R. The maximum norm of a is defined as $\|a\|_\infty = \max_i\{|a_i|\}$. Let χ be a probability distribution on R. Assume an efficient sampler for elements of R according to χ and use the notation $a \leftarrow \chi$ to denote that $a \in R$ is sampled from χ. The distribution χ on R is called B-bounded from some $B > 0$ if for all $a \leftarrow \chi$ we have $\|a\|_\infty < B$, i.e., a is B-bounded. As an example, let $\mathcal{D}_{\mathbb{Z},\sigma}$ be the discrete Gaussian over the integers with mean 0 and standard deviation σ, which assigns a probability proportional $\exp(-\pi|x|^2/\sigma^2)$ to each $x \in \mathbb{Z}$. When d is a power of 2, whence $\Phi_d(X) = X^n + 1$, we can take χ to be the spherical discrete Gaussian $\chi = \mathcal{D}_{\mathbb{Z}^n,\sigma}$ where each coefficient of the polynomial is sampled according to the one-dimensional distribution $\mathcal{D}_{\mathbb{Z},\sigma}$.

In addition to the ring R we require the ring R_q which is simply the set of polynomials R taken modulo q. We denote the map that reduces an integer x modulo q and uniquely represents the result by an element in the interval $(-q/2, q/2]$ by $[\cdot]_q$. We extend this map to polynomials in $\mathbb{Z}[X]$ and thus also to elements of R by applying it to their coefficients separately, i.e., $[\cdot]_q : R \leftarrow R$, $a = \sum_{i=0}^{n-1} a_i X^i \mapsto \sum_{i=0}^{n-1} [a_i]_q X^i$. Furthermore we extend this notation to vectors of polynomials by applying to the entries of the vectors separately. A polynomial $f \in R$ is invertible modulo q if there exists a polynomial $f^{-1} \in R$ such that $ff^{-1} = \tilde{f}$, where $\tilde{f}(X) = \sum_i a_i X^i$ with $a_0 = 1 \pmod q$ and $a_j = 0 \pmod q$ for all $j \neq 0$. In addition to the modulus q that is used to reduce the coefficients of the elements that represent ciphertexts, there is a second modulus $t < q$ that determines the message space $R_t = R/tR$. Let B_{err} be the bound on the support of the truncated Gaussian, χ_{err}, from which noise is sampled.

For two strings x and y over an alphabet Σ, we denote the Hamming distance of x and y by $\Delta_\Sigma(x, y)$. In the case that $\Sigma = \{0, 1\}$, we drop the subscript and simply denote the distance by $\Delta(x, y)$. The Hamming weight of a binary string x is the number of ones in x, or equivalently $\Delta(x, 0^n)$, we denote this quantity by $|x|$. The Hamming ball of radius D around a string x is defined as $\{y : \Delta_\Sigma(x, y) < D\}$ and is denoted $B_\Sigma(x, D)$. For a set X, by abuse of notation we define $B_\Sigma(X, D) = \cup_{x \in X} B_\Sigma(x, D)$.

2.1 Levelled Homomorphic Encryption

We recall the more practical variant of the levelled homomorphic encryption scheme YASHE [12].

Fix a word size w. Then, an element $x \in R$ with coefficients in $(-q/2, q/2]$ can be written as $\sum_{i=0}^{\ell_{w,q}-2} [x_i]_w w^i$ where $[x_i]_w \in (-w/2, w/2]$ and $\ell_{w,q} = \lfloor \log_w(q) \rfloor + 2$. The scheme follows.

- $\mathsf{ParamsGen}(\lambda)$: Given the security parameter λ, fix a positive integer d that determines R, moduli q and t with $1 < t < q$, and distributions $\chi_{\text{key}}, \chi_{\text{err}}$ on R. Output $(d, q, t, \chi_{\text{key}}, \chi_{\text{err}})$.
- $\mathsf{Keygen}(d, q, t, \chi_{\text{key}}, \chi_{\text{err}})$: Sample $f', g \leftarrow \chi_{\text{key}}$ and let $f = [tf' + 1]_q$. If f is not invertible modulo q, choose a new f'. Compute the inverse $f^{-1} \in R$ of f modulo q and set $h = [tgf^{-1}]_q$. Output $(\mathsf{pk}, \mathsf{sk}) = (h, f)$.
- $\mathsf{Enc}(h, m)$: For message $m + tR$, sample $s, e \leftarrow \chi_{\text{err}}$ and output ciphertext $[\lfloor q/t \rfloor [m]_t + e + hs]_q \in R$.

- $\mathsf{Enc}^*(h, m)$: For message $m + tR$, output ciphertext $[\lfloor q/t \rfloor [m]_t]_q \in R$.
- $\mathsf{Dec}(f^s, c)$: To decrypt ciphertext c, compute

$$[\lfloor \frac{t}{q} \cdot [f^s c]_q \rceil]_t \in R$$

- $\mathsf{Add}(c_1, c_2)$: Output $[c_1 + c_2]_q$.
- $\mathsf{Mult}(c_1, c_2)$: Output the ciphertext $\tilde{c}_{\text{mult}} = [\lfloor \frac{t}{q} c_1 c_2 \rceil]_q$

For ciphertexts $c_1, c_2 \in R$ that encrypt $m_1, m_2 \in R$, the ciphertext $\tilde{c}_{\text{mult}}$ during homomorphic multiplication satisfies $f^2 = \tilde{c}_{\text{mult}} = \Delta[m_1 m_2]_t + \tilde{v}_{\text{mult}} \bmod q$, this implies that $\tilde{c}_{\text{mult}}$ is an encryption of $[m_1 m_2]_t$ under f^2.

Batch representation. In the case that $q \equiv 1 \pmod t$ and $t \equiv 1 \pmod{2n}$, for prime t, the Chinese remainder theorem yields

$$\frac{\mathbb{Z}_t[X]}{(X^n + 1)} \cong \prod_{i=1}^{n} \frac{\mathbb{Z}_t[X]}{(Q_i(X))} \pmod t$$

where $Q_i(X)$ are linear polynomials. This yields an efficient map which takes n elements from the right hand side and produces a single plaintext polynomial. We denote this map by CRT.

Selecting Parameters. Let λ be the security parameter. Let q be the coefficient modulus, i.e., the modulus used to reduce the coefficients of ciphertexts. Let χ_{key} be the uniform distribution of length $2n$ strings over $\{-1, 0, 1\}$ and let χ_{err} be the discrete Gaussian $\mathcal{D}_{\mathbb{Z},\sigma}$ with maximal deviation from the mean B_{err}, where $B_{err} > \sigma\sqrt{\lambda}$. In that case the following bounds hold with $1 - \text{negl}(\lambda)$ probability, (see Appendix K [12] and the refinements in Appendix A [13]). In all cases let v_1 and v_2 be the inherent noise [12, 13] associated with input ciphertexts c_1 and c_2 and let v be the inherent noise associated with the output c produced by the respective homomorphic operation.

- Encryption: $\|v\|_\infty < 2tn^{1/2} B_{err}$
- Addition: $\|v\|_\infty < \|v_1\|_\infty + \|v_2\|_\infty + t$
- Multiplication without relinearization: $\|v\|_\infty < \frac{t^2}{2} n^{3/2} (\|v_1\|_\infty + \|v_2\|_\infty)$
- Addition by plain: $\|v\|_\infty$
- Multiplication by plain: $\sqrt{\deg(p) + 1} \|v\|_\infty \|p\|_\infty$
- Negation: $\|v\|_\infty$

These bounds are used to determine correctness, while semantic security of the scheme follows from the following assumptions [17, 33] after taking an appropriately large coefficient modulus q, relative to the lattice dimension n.

Definition 1 (Decision-RLWE Assumption)**.** *Given security parameter λ, let d and q be integers depending on λ, $R = \mathbb{Z}[X]/(\Phi_d(X))$ and $R_q = R/qR$. Given a distribution χ*

over R_q that depends on λ, the Decision-$\mathsf{RLWE}_{d,q,\chi}$ *problem is to distinguish the following two distributions. The first distribution consists of pairs (a, u), where $a, u \leftarrow R_q$ are drawn uniformly at random from R_q. The second distribution consists of pairs of the form $(a, a \cdot s + e)$. The element $s \leftarrow R_q$ is drawn uniformly at random and is fixed for all samples. For each sample $a \leftarrow R_q$ is drawn uniformly at random and $e \leftarrow \chi$. The* Decision-$\mathsf{RLWE}_{d,q,\chi}$ *assumption is that the* Decision-$\mathsf{RLWE}_{d,q,\chi}$ *problem is hard.*

Definition 2 (Decision-SPR Assumption)**.** *For security parameter λ, let d and q be integers, $R = \mathbb{Z}[X]/(\Phi_d(X))$, $R_q = R/qR$ and χ be a distribution over R_q, all depending on λ. Let $t \in R_q^\times$ be invertible in R_q, $y_i \in R_q$ and $z_i = -y_i t^{-1} \bmod q$ for $i \in \{1, 2\}$. The* Decision-$\mathsf{SPR}_{d,q,\chi}$ *problem is to distinguish elements of the form $h = a/b$ where $a \leftarrow y_1 + t \cdot \chi_{z_1}, b \leftarrow y_2 + t \cdot \chi_{z_2}$ from uniformly random elements of R_q. The* Decision-$\mathsf{SPR}_{d,q,\chi}$ *assumption is that the* Decision-$\mathsf{SPR}_{d,q,\chi}$ *problem is hard.*

3 Problems

We consider three tasks involving private matching of k-mers held by a consumer (client) and k-mers held by a provider (server). In all cases, the threat model we consider in the main body of our work is that of static semi-honest adversaries, while modifications to achieve security against malicious adversaries are detailed in the Appendix. We assume that the cardinalities of the client set and server set may be shared and are effectively public information. We use the notation of Hazay and Lindell [34] to describe the two-party ideal functionality in each scenario. For our purposes a k-mer is a string of length k over the alphabet $\Sigma = \{A, C, G, T\}$.

3.1 Scenarios

Scenario 1. The client holds a set of k-mers X, while the server holds a set of k-mers Y. Associated with each k-mer in the server set is a label holding diagnostic information, this provided by a labelling function $\ell(\cdot)$. The problem is for the client to learn the labels associated with the k-mers in common with the server set. Nothing else should be revealed by the computation, in particular the server set and associated labels not in the intersection must remain private. Formally we wish to compute the functionality

$$(X, (Y, \ell(\cdot))) \to (\ell(y) : y \in X \cap Y, \lambda)$$

Scenario 2. As in Scenario 1, the client holds a set of k-mers X, while the server holds a set of k-mers Y. In this case, however, exact-matching is not practicable, and a tolerance for error must be allowed. We model this as matching those k-mers in the client set within Hamming distance D of the server set, over the alphabet Σ. As in Scenario 1, there is a set of labels associated with server k-mers, and the problem is for the client to learn the labels associated with those k-mers in the server set for which a fuzzy match with the client set exists. As in Scenario 1, the server set and associated labels must remain private. Additionally we assume the threshold distance used may be provider specific and thus should also remain hidden. Formally we wish to compute the functionality

$$(X, (Y, \ell(\cdot), D)) \to (\ell(y) : y \in B_\Sigma(X, D), \lambda)$$

Scenario 3. In this task, the server holds a set of k-mers Y, while the client holds a single k-mer x. The client wishes to compute a match between x and the server set Y, however as in Scenario 2, exact matching is not possible. Therefore the problem is for the client to compute the k-mer in Y with smallest Hamming distance from x. As in Scenarios 1 and 2, the server set must remain private. Formally, we wish to compute the functionality

$$(x, Y) \to (\arg\min_{y \in Y} \Delta_\Sigma(x, y), \lambda)$$

These functionalities are designed for the specific genomics problems described in the introduction, though the use of a generic labelling function in Scenarios 1 and 2 allows significant flexibility in accommodating distinct but related tasks. As an example, if $\ell(\cdot)$ is taken not to be a function providing diagnostic information, but instead to be the identity function, one may compute functionalities corresponding to private intersection of X and Y, and the private intersection of X and Hamming ball around Y of radius D.

3.2 Representation

We represent individual k-mers using one of two representations as follows.

Two-bit-base. Exactly two bits are used to represent each nucleotide. Therefore each k-mer is efficiently represented as $2k$-bit string.

Indicator variable. Indicator variables are used to represent each nucleotide, yielding four orthogonal bit strings. Therefore each k-mer is efficiently represented as a $4k$-bit string of Hamming weight k.

The first representation leads to a concise representation of consumer/provider sets as sets of strings in $\{0, 1\}^{2k}$. The latter is much more convenient for computing Hamming distances, in particular the weight of two xor-ed k-mers in this representation is exactly twice the corresponding Hamming distance over Σ.

4 Set Intersection

In this section, we describe an efficient solution for Scenario 1 using batched homomorphic cryptography. We first describe a protocol for set membership, before showing how to adapt this protocol to set intersection, and finally to set intersection with labelling.

$$A = \begin{bmatrix} a_0, & \cdots & , a_0 \\ a_1, & \cdots & , a_1 \\ \vdots, & \vdots & , \vdots \\ a_{|X|-1}, & \cdots & , a_{|X|-1} \end{bmatrix} \quad B = \begin{bmatrix} y_1^0, & \cdots & , y_n^0 \\ y_1^1, & \cdots & , y_n^1 \\ \vdots, & \vdots & , \vdots \\ y_1^{|X|-1}, & \cdots & , y_n^{|X|-1} \end{bmatrix}$$

Figure 1: Matrices A and B used to evaluate client polynomial $P(\cdot)$ on a batch of server plaintexts $y_1, \ldots, y_n$.

The starting point for our private set intersection protocol is that described in Section 4.1 [10]. In this protocol the client computes a polynomial P which vanishes on their input set. The coefficients of its polynomial, $(a_0, \ldots, a_{|X|-1})$ are sent in encrypted form to the server. The server then homomorphically computes $\sum_{i=0}^{|X|-1} \mathsf{Enc}(a_i) \ltimes y^i = \mathsf{Enc}(P(y))$. Let u be a random scalar. Then $\mathsf{Enc}(P(y)) \ltimes u + \mathsf{Enc}(y)$ is an encryption of y if y is an element of X, otherwise a uniformly distributed element y' otherwise. Our challenges are two-fold, firstly we need to transform this protocol into one which operates on batches of set elements, and secondly we have to accommodate the transference of labels as described in Scenario 1, Section 3.

The first problem we solve as follows. Given the matrices A and B described in Figure 1, the homomorphic row sum of the component wise product $A \circ B$ results in an encryption of the vector $P(y_1), \ldots, P(y_n)$ for a chunk of n elements $y_1, \ldots, y_n \in Y$. On the other hand, this product is readily computed by SIMD multiplication of the corresponding rows of A and B followed by $|X|$ many SIMD sums. By looping over distinct chunks of n elements of Y at a time, we can efficiently determine membership in X of every element in Y. To solve the second problem, one may conveniently transfer a label ℓ rather than k-mer y, by computing $\mathsf{Enc}(P(y)) \ltimes u + \ell$, where ℓ is contained in $2k$ bits and is therefore more than ample for this problem. The SIMD form of this scales the vector of evaluations $(P(y_i))$ with a vector of randomizers followed by a translation by the vector of corresponding labels.

4.1 Protocol Description – Scenario 1

Using the two-bits-per base representation, we may assume that all k-mers can be embedded into a plaintext modulus, t, greater than 2^{2k}. All algebraic operations take place modulo t.

The client begins by encoding their set as a polynomial P of degree $|X|$ with coefficients $a_0, \ldots, a_{|X|-1}$. The client then generates a sequence of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_{|X|}$ where the i^{th} ciphertext is a batch encryption of $(a_{i-1}, \ldots, a_{i-1})$. These ciphertexts are sent to the server.

The server then splits their private set into a sequence of batches $1, \ldots, I$. For the i^{th} batch, $y_{(i-n)+1}, \ldots, y_{in}$, the server generates plaintext polynomials $(p_i^{(j)})_{j \in [|X|]}$ where $p_i^{(j)}$ encodes the sequence $y_{(i-n)+1}^{j-1}, \ldots, y_{in}^{j-1}$, i.e., $(j-1)^{th}$ powers of k-mers in that batch. It also a prepares a label vector p_i^{ℓ} corresponding to the labels of k-mers in this batch, as well as a randomizer vector p_i.

For each i, the server computes $\sum_{j=1}^{|X|} \mathsf{ct}_i \ltimes p_i^{(j)}$, which due to the SIMD properties of encryption is equivalent to evaluating the client polynomial on this batch of k-mers. This ciphertext is finally blinded with p_i and translated by q_i.

---

procedure SET-INTERSECTION-I$(X, Y, \ell(\cdot))$
  $\mathbf{P_1}, \mathbf{P_2}$: $I \leftarrow \frac{|Y|}{n}, t \leftarrow p_{\geq 2^{2k}}$
  $\mathbf{P_1}$:
    Enumerate X as $\{x_1, \ldots, x_{|X|}\}$
    Construct $P(\cdot) = \sum_{i=0}^{|X|-1} a_i x^i$ such that
    $P(x_i) = 0$ for $i = 1, \ldots, |X|$.
    for $i \leftarrow 1$ to $|X|$ **do**
      $\mathsf{ct}'_i \leftarrow \mathsf{Enc}(\mathsf{CRT}(a_{i-1}, \ldots, a_{i-1}), \mathsf{pk})$
    end for
    Send $(\mathsf{ct}'_1, \ldots, ct'_{|X|})$.
  $\mathbf{P_2}$:
    Enumerate Y as $\{y_1, \ldots, y_{|Y|}\}$
    for $i \leftarrow 1$ to I **do**
      for $j \leftarrow 1$ to $|X|$ **do**
        $p_i^{(j)} \leftarrow \mathsf{CRT}(y_{(i-1)n+1}^{j-1}, \ldots, y_{in}^{j-1})$
      end for
      Pick $u_{i1}, \ldots, u_{in} \in_R \mathbb{Z}_t$
      $p_i \leftarrow \mathsf{CRT}(u_{i1}, \ldots, u_{in})$
      $q_i \leftarrow \mathsf{CRT}(\ell(y_{(i-1)n+1}), \ldots, \ell(y_{in}))$
      $\mathsf{ct}_i \leftarrow (\sum_{j=1}^{|X|} \mathsf{ct}'_i \ltimes p_i^{(j)}) \ltimes p_i + q_i$
    end for
    Send $(\mathsf{ct}_1, \ldots, \mathsf{ct}_I)$.
end procedure

Lemma 1. SET-INTERSECTION-I *correctly computes set intersection with labelling if* $8 \cdot |X| \cdot 2^{6k+3} \cdot n^{3/2} \cdot B_{err} < q$ *except with* $\mathrm{negl}(\lambda)$ *probability.*

Proof. Let n_{fresh} be the noise in a fresh ciphertext. We have that, for $i \leq \frac{|Y|}{n}$, ciphertext ct_i is computed as a sum of $|X|$ terms, each with noise $n_{\text{fresh}} \cdot t \cdot n^{1/2}$, yielding noise $|X| \cdot n_{\text{fresh}} \cdot t \cdot n^{1/2}$. This term then incurs a final multiplicative factor of $t \cdot n^{1/2}$, yielding total noise $4 \cdot |X| \cdot t^3 \cdot n^{3/2} \cdot B_{err}$. This term is bounded by $\frac{q}{2}$ (minus a small term we may safely ignore), yielding the result. □

Lemma 2. SET-INTERSECTION-I *achieves security against statically chosen semi-honest adversaries if the* Decision-$\mathsf{LWE}_{2n,q,\chi}$ *and* Decision-$\mathsf{SPR}_{2n,q,\chi}$ *assumptions hold.*

Proof. Simulator $\mathcal{S}$ proceeds as follows. Generate $|X|$ random ciphertexts and forwards them to P_2. On receipt of the ciphertexts by P_2, $\mathcal{S}$ generates $\frac{|Y|}{n}$ random ciphertexts and forwards them to P_1. By a hybrid argument for any PPT adversary $\mathcal{A}$ with advantage ϵ distinguishing the real protocol from the simulation, we can construct a PPT adversary $\mathcal{A}'$ with advantage at least $\frac{\epsilon}{|X|+|Y|/n}$ in breaking either the Decision-$\mathsf{LWE}_{2n,q,\chi}$ assumption or the Decision-$\mathsf{SPR}_{2n,q,\chi}$ assumption. Since $|X|$ and $|Y|$ are constants, the claim follows. □

In Appendix A we describe a private set intersection scheme with quasi-linear complexity based upon the multi-round protocol of Kissner and Song [11].

5 Fuzzy Matching

In this section, we describe an efficient scheme for matching the set of k-mers submitted by a patient to that of a reference set provided by the lab when exact matching is not possible.

The difficulty in solving this problem is that a naïve solution based upon homomorphic evaluation of the binary circuit computing fuzzy matching does not necessarily yield a very efficient solution when translated to homomorphic computation. To see this, observe that computing the Hamming distance between two $4k$-bit strings requires $O(\log k)$ depth as a binary circuit. Not only does this incur logarithmic overhead in homomorphic operations but the circuit is not very amenable to SIMD arithmetic operations. To solve this problem we consider augmenting a ciphertext corresponding to a set element x with encryptions of every possible cyclic shift of x. A similar expansion was previously used [35, 36] in the context of cryptographic counters for preferential voting. Given this expanded list, computing the Hamming distance homomorphically is possible using the observation $\sum_{j=1}^{4k}(x^{(j)} \oplus y^{(j)}) = \Delta(x,y)^{4k}$, where $x^{(j)}$ and $y^{(j)}$ correspond to x and y cyclically shifted j positions respectively i.e., the component-wise sum over the integers of all cyclic shifts of x and y xor-ed yields a vector containing $4k$ copies of $\Delta(x,y)$. Recall from Section 3, that x and y are in indicator variable format, thus the binary Hamming distance $\Delta(x,y)$ is twice the corresponding distance over the alphabet $\Sigma = \{A, C, G, T\}$. It follows that $\mathsf{Enc}(\sum_{j=1}^{4k}(x^{(j)} \oplus y^{(j)})) - \mathsf{Enc}(0, 2, \ldots, 2(D-1), 0, \ldots, 0)$ is a ciphertext containing a zero iff $\Delta_\Sigma(x,y) < D$, thus one can transfer a label ℓ_y corresponding to y using the "affine" trick on the ciphertext, described in the previous section. By batch encrypting cyclic shifts of elements of X and homomorphically summing against the corresponding cyclic shifts of an element y we get an efficient fuzzy membership protocol for y. Iterating over every element in the server set thus yields an efficient fuzzy matching protocol. One wrinkle we have ignored in this description is what happens when a label ℓ_y does not fit in base field provided by the plaintext modulus, which is typically small and therefore likely inadequate for embedding of meaningful labels. We solve this problem by replacing the comparison vector with $(0, 2, \ldots, 2(D-1), 0, 2, \ldots)$, i.e., repeating the target distance set sequentially until all slots in the target ciphertext are filled. This enables us to cram $\frac{n}{D}$ zeroes into the differenced ciphertext, thus with appropriate dissection of labels to fit individual slots – and assuming n greater than k^2, one can transfer a label of at least k bits.

5.1 Protocol Description – Scenario 2

Using indicator variable format, we may assume that all k-mers are represented by strings of $4k$ bits. To compute Hamming weights/distances in this representation we require a plaintext modulus counting even values up to and including $2k$.

The client first splits their k-mer set into a sequence of batches $1, \ldots, I$. Let $n' = \frac{n}{4k}$. The i^{th} batch of k-mers $x_{(i-1)n'+1}, \ldots, x_{in'}$ is encoded as a ciphertext $\mathsf{ct}_i^{(1)}$ as well as all the sequences of k-mers shifted j positions, $x^{(j)}_{(i-1)n'+1}, \ldots, x^{(j)}_{in'+1} : j \in [4k]$, which are encoded in $\mathsf{ct}_i^{(2)}, \ldots, \mathsf{ct}_i^{(4k)}$. These ciphertexts are sent to the server.

For each k-mer y in their set, the server will perform the following operations. The server generates plaintext polynomials $(p_y^{(j)})_{j \in [4k]}$ encoding the j^{th} shift of y duplicated n' times. The server generates a target vector containing the set of possible Hamming distances $\{0, 2, \ldots, 2(D-1)\}$ duplicated to fill up all n slots in the ciphertext. The server generates label vector ℓ_y and randomizer vector p_i.

For the i^{th} batch of received ciphertexts, the server computes $\sum_{j=1}^{4k}(\mathsf{ct}_i^{(j)} - p_y^{(j)})^2$ this computes a vector containing $\Delta(x_{(i-1)n'+1}, y)^{4k} \| \ldots \| \Delta(x_{in'+1}, y)^{4k}$. The target vector is subtracted and the result then blinded with p_i and then as usual, translated by p_{ℓ_y}.

---

procedure Fuzzy-Matching$(X, Y, \ell(\cdot), D)$
  $\mathbf{P_2}$: Enumerate Y as $\{y_1, \ldots, y_{|Y|}\}$
  for $i \leftarrow 1$ to $|Y|$ **do**
    $\mathbf{P_2}$: Fuzzy-Membership$(X, y_i, \ell(y_i), D)$
  end for
end procedure

One can transfer a matching k-mer of $\theta(k)$ bits directly by replacing the plaintext modulus with a prime greater than $2^{\theta(k)/4}$ and taking $\ell(\cdot)$ to be the identity function.

In addition to transferring matching k-mers rather than labels, it is also possible to solve the problem when the target distances are taken from an arbitrary set S rather than a threshold set. The solution is no more expensive than the threshold case and involves replacing the target plaintext with the set $2 \cdot S$ as a vector, repeated sequentially to fill all slots.

Lemma 3. Fuzzy-Matching *correctly computes fuzzy matching with labelling if* $2^{11} \cdot k \cdot n^{13/2} \cdot \log^4 n \cdot B_{err} < q$ *except with* $\mathrm{negl}(\lambda)$ *probability.*

Proof. We have that, for $i \leq \frac{4k \cdot |X|}{n}$, ciphertext ct_i is computed as the sum of $4k$ terms, where each term has noise exactly equivalent to the product of two freshly encrypted ciphertexts (the subtractions by cleartext terms leave the noise unchanged). Considering the noise contribution in each product, the noise in this sum is thus $4k \cdot (2 \cdot n^{3/2} \cdot t^2 \cdot n_{\text{fresh}})$. Considering the multiplication by randomiser p_i, this term incurs a final multiplicative factor of $t \cdot n^{1/2}$. In all, we have that $16 \cdot k \cdot n^{5/2} \cdot t^4 \cdot B_{err}$ should be bounded by $\frac{q}{2}$. Finally, the Prime Number Theorem implies that $t \sim 2n \log 2n$ to achieve $t \equiv 1 \bmod 2n$, completing the proof.

procedure FUZZY-MEMBERSHIP$(X, y, \ell_y, D \in [k+1])$

- $\mathbf{P_1}, \mathbf{P_2} : n' \leftarrow \frac{n}{4k}, I \leftarrow \frac{|X|}{n'}, t \sim 2n \log 2n$
- $\mathbf{P_1}$:
  - Enumerate X as $\{x_1, \ldots, x_{|X|}\}$
  - **for** $i \leftarrow 1$ to I **do**
    - **for** $j \leftarrow 1$ to $4k$ **do**
      - $\mathsf{ct}_i^{(j)} \leftarrow \mathsf{Enc}(\mathsf{CRT}(\underbrace{x^{(j)}_{(i-1)n'+1} \| \ldots \| x^{(j)}_{in'}}_{n'}), \mathsf{pk})$
    - **end for**
    - Send $(\mathsf{ct}_i^{(1)}, \ldots, \mathsf{ct}_i^{(4k)})$.
  - **end for**
- $\mathbf{P_2}$:
  - **for** $j \leftarrow 1$ to $4k$ **do**
    - $\mathsf{ct}_y^{(j)} \leftarrow \mathsf{Enc}^*(\mathsf{CRT}(\underbrace{y^{(j)} \| \ldots \| y^{(j)}}_{n'}), \mathsf{pk})$
  - **end for**
  - $\mathsf{ct}_{\mathrm{tgt}} \leftarrow \mathsf{Enc}^*(\mathsf{CRT}($
  - $\underbrace{0, \ldots, 2(D-1), 0, \ldots, 2(D-1), 0, 2, \ldots}_{n}), \mathsf{pk})$
  - Write $\ell_y = \ell_1 \| \ldots \| \ell_v$ where $v = \lfloor \frac{4k}{D} \rfloor$
  - $\mathsf{ct}_{\ell_y} \leftarrow \mathsf{Enc}^*(\mathsf{CRT}($
  - $\underbrace{\overbrace{\ell_1, \ldots, \ell_1}^{D}, \ldots, \overbrace{\ell_v, \ldots, \ell_v}^{D}, \overbrace{0, \ldots, 0}^{4k-vD} \overbrace{\ell_1, \ldots, \ell_1}^{D}, \ldots}_{n}$
  - $), \mathsf{pk})$
  - **for** $i \leftarrow 1$ to I **do**
    - Pick $u_{i1}, \ldots, u_{in} \in_R \mathbb{Z}_t$
    - $p_i \leftarrow \mathsf{CRT}(u_{i1}, \ldots, u_{in})$
    - $\mathsf{ct}_i \leftarrow (\sum_{j=1}^{4k} (\mathsf{ct}_i^{(j)} - \mathsf{ct}_y^{(j)})^2 - \mathsf{ct}_{\mathrm{tgt}}) \ltimes p_i + \mathsf{ct}_{\ell_y}$
  - **end for**
  - Send $(\mathsf{ct}_1, \ldots, \mathsf{ct}_I)$.

end procedure

□

Lemma 4. Fuzzy-Matching *achieves security against statically chosen semi-honest adversaries if the* Decision-$\mathsf{LWE}_{2n,q,\chi}$ *and* Decision-$\mathsf{SPR}_{2n,q,\chi}$ *assumptions hold.*

Proof. Simulator $\mathcal{S}$ proceeds as follows. Generate $\frac{16k^2|X|}{n}$ random ciphertexts and forwards them to P_2. On receipt of the ciphertexts by P_2, $\mathcal{S}$ generates $\frac{4k|Y|}{n}$ random ciphertexts and forwards them to P_1. By a hybrid argument for any PPT adversary $\mathcal{A}$ with advantage ϵ distinguishing the real protocol from the simulation, we can construct a PPT adversary $\mathcal{A}'$ with advantage at least $\frac{\epsilon}{16k^2|X|/n+4k|Y|/n}$ in breaking either the Decision-$\mathsf{LWE}_{2n,q,\chi}$ assumption or the Decision-$\mathsf{SPR}_{2n,q,\chi}$ assumption. Since $|X|$ and $|Y|$ are constants, the claim follows. □

6 Closest Matching

One possible approach to solving this problem is to modify the protocol for fuzzy matching in the previous section to transfer k-mers from the provider set, rather than their labels (this is readily achieved by the remarks in the bottom paragraph of the preceding section). One then iterates this protocol with increasing threshold distance, until a match is found. Although this approach has the advantage of following from our protocol, or indeed any other protocol for private fuzzy matching, it does not meet our goal of low overall round complexity corresponding to the online/offline paradigm described in the introduction. Consequently we abandon it in favor of other approaches.

For our second attempt, we observe that finding a closest match is equivalent to computing Hamming weights of xor-ed client/provider k-mers, and then comparing these weights until a minimum is found. Unfortunately it is far from clear either a) how to compute these weights b) how to compare them in encrypted form. For the second problem, we turn to encoding integer weights as indicator vectors. The advantage of this representation is that it allows comparison to be computed homomorphically in constant depth rather than the linear depth that would be required by such comparison on binary strings. Specifically, given indicator vectors W_1, W_2 encoding integer weights w_1, w_2 in the interval $[0, 2k+1)$, one first enumerates all pairs $(W_1[i], W_2[j])$ where i and j are indices in $[2k+1]$ and where $i > j$. It can be seen that the sum $\Sigma_{i>j} W_1[i] \cdot W_2[j]$, evaluates the function $w_1 > w_2$, i.e., the comparison operation.

With this format, let's see how the approach of Alperin-Sheriff *et al.* [32] for computing addition by rotation matrices can be adapted to solve the first problem of determining Hamming weights of the xor-ed k-mers. Let the bits of the client k-mer be $b_1, \ldots b_{4k}$. In that case, the client can send permutation matrices $(P_{j0}, P_{j1})_{j\in[4k+1]}$ of dimension $2k+1$ corresponding to shifts $b_j, 1-b_j$ in $\mathbb{Z}_{2k+1}$. Now the server, for each of their k-mers y, computes the product $\prod_{j=1}^{4k} P_{j,y[j]}$ corresponding to the shift $\sum_{j=1}^{4k}(b_j \oplus y[j]) \in \mathbb{Z}_{2k+1}$, which is in fact the Hamming weight of $x \oplus y$.

A major drawback of this approach however is that $O(k)$ aggregate multiplication operations are required to compute the result leading to an exponential noise level which is infeasible to decrypt. Fortunately we can overcome this obstacle. The solution turns out to be for the client to first split their k-mer into $O(\sqrt{k})$ chunks each of $O(\sqrt{k})$ bits, then for every possible xor-mask of each chunk, the client computes the Hamming weight of corresponding masked chunk. These "sub-weights" are sent in encrypted indicator vector format to the server. For each k-mer y, the server splits it into chunks $y^{(1)}, \ldots, y^{(O(\sqrt{k})}$ and is able to compute the total Hamming weight of $x \oplus y$ by retrieving the weight corresponding to $x^{(j)}$ xor-ed with $y^{(j)}$ for each j and summing these weights using the usual expand-and-multiply procedure. This entails multiplying only $O(\sqrt{k})$ permutation matrices, thus the noise is reduced to $O(\sqrt{k})$ aggregate multiplications.

6.1 Protocol Description – Scenario 3

The client begins by splitting their query k-mer into $M = \sqrt{4k}$ chunks, $x^{(1)}, \ldots, x^{(M)}$ each of $\frac{4k}{M}$ bits. Now for every possible provider mask of M bits, msk, the client computes the Hamming weight of $x^{(j)} \oplus \mathrm{msk}$ and stores the result as an encrypted indicator vector of length $M + 1$. The list of encrypted weights corresponding to each chunk is sent to the server.

The server, for every k-mer y_i in their set, retrieves the encrypted indicator vectors corresponding to $\mathrm{msk}_1 = y_i^{(1)}, \ldots, \mathrm{msk}_M = y_i^{(M)}$. These vectors are expanded into rotation matrices $P_{i1}, \ldots, P_{iM}$ of dimension $2k + 1$. The product of these matrices yields a single rotation matrix P_i corresponding to the Hamming weight of $y_i \oplus x$. The first column, of this matrix, W_i contains this weight in indicator vector format. Associated to this vector is the encryption of the corresponding k-mer, encrypted in batch format for efficiency, which we denote ct_i.

To find the closest match we need a comparison function between different weight vectors. For inputs W and W' this is effected by summing over the products $W[a] \cdot W'[b] : a > b, a, b \in \{0, 2, \ldots, 2k\}$.

Given this homomorphic comparison function, finding the closest match is possible simply by imposing a binary tree on the array of pairs (ct_i, W_i) and searching pair-wise from leaves to root using this procedure. This entails an overhead of $\log_2 |Y|$ aggregate multiplication operations.

---

procedure EXPAND($W, 2k + 1$)

$$P \leftarrow \left.\overbrace{\begin{bmatrix} W^T & 0 & \cdots & 0 \\ 0 & W^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^T \end{bmatrix}}^{2k+1}\right\}\text{2k+1}$$

return P

end procedure

In general one can efficiently trade off space/time by changing the number of chunks M to $(4k)^c$ for a constant $0 < c < 1$. A larger value of c requires a larger data transfer but

procedure CLOSEST-MATCHING(x, Y)

  $\mathbf{P_1}, \mathbf{P_2}$: $M \leftarrow \sqrt{4k}, t \leftarrow 2$

  $\mathbf{P_1}$:

    Write $x = x^{(1)} \| \dots \| x^{(M)}$ with $|x^{(j)}| = \frac{4k}{M}$

    for $j \leftarrow 1$ to M **do**

      for $\text{msk} \in \{0,1\}^{\frac{4k}{M}}$ **do**

        $I \leftarrow \mathbb{1}_{|x^{(j)} \oplus \text{msk}|}$

        $W_{j,\text{msk}} \leftarrow$

        $(\mathsf{Enc}(I[0], \mathsf{pk}), \dots, \mathsf{Enc}(I[\frac{4k}{M}], \mathsf{pk}))$

      end for

    end for

    Send $(W_{j,\text{msk}})_{j \in 1+[M], \text{msk} \in \{0,1\}^{\frac{4k}{M}}}$.

  $\mathbf{P_2}$:

    Enumerate Y as $\{y_1, \dots, y_{|Y|}\}$

    for $i \leftarrow 1$ to $|Y|$ **do**

      Write $y_i = y_i^{(1)} \| \dots \| y_i^{(M)}$

      for $j \leftarrow 1$ to M **do**

        $P_{ij} \leftarrow \text{EXPAND}(W_{j, y_i^{(j)}}, 2k+1)$

      end for

      $P_i \leftarrow \prod_{j=1}^{M} P_{ij}$

      $W_i \leftarrow P_i[1]$

      $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{CRT}(y_i[1], \dots, y_i[4k]), \mathsf{pk})$

    end for

    $N \leftarrow \frac{|Y|}{2}$

    for $h \leftarrow 0$ to $\log_2 |Y| - 1$ **do**

      for $i \leftarrow 1$ to N **do**

        $\mathsf{ct}_{(<)} \leftarrow \sum_{(a,b):b>a} W_i[a] \cdot W_{i+2^h}[b]$

        $\mathsf{ct}_i \leftarrow \mathsf{ct}_{i+2^h} + \mathsf{ct}_{(<)} \cdot (\mathsf{ct}_i - \mathsf{ct}_{i+2^h})$

        $W_i \leftarrow W_{i+2^h} + \mathsf{ct}_{(<)} \cdot (W_i - W_{i+2^h})$

        $N \leftarrow \frac{N}{2}$

      end for

    end for

    Send ct_1.

end procedure

results in less time to find the match and accommodates an encryption scheme with lower noise tolerance.

Lemma 5. Closest-Matching *correctly computes closest matching if* $8 \cdot (4k)^{(1-c)(4k)^c} \cdot (k+1)^{2\log_2|Y|} \cdot (4 \cdot n^{3/2})^{(4k)^c + 2\log_2|Y|+1} \cdot B_{err} < q$ *except with* $\mathrm{negl}(\lambda)$ *probability.*

Proof. We analyze the cumulative noise in the weight vector W_1, associated to the closest matching ciphertext ct_1, in two phases. In the first phase we compute the noise incurred by multiplying the M rotation matrices (recall that W_1 is taken as the first column of P_1). In the second phase, we compute the additional noise on W_1 incurred by the tree search. Let n_l be the maximum noise in any ciphertext in the partial product $P_l =: \prod_{j=1}^{l} P_{1j}$. Since P_{l+1} is computed as the dot product of up to $\frac{4k}{M}$ ciphertexts of noise level n_l and $\frac{4k}{M}$ ciphertexts of noise level n_{fresh}, we have that $n_{l+1} = a \cdot (n_l + n_{\text{fresh}}))$ where $a = \frac{4k}{M} \cdot n^{3/2} \cdot t^2$. Solving this recurrence yields $n_M = a^{M-1} \cdot n_1 + \sum_{l=1}^{M-1} a^l \cdot n_{\text{fresh}}$. As $n_1 = n_{\text{fresh}}$, we have $n_M < 2 \cdot a^M \cdot n_{\text{fresh}}$. For the second phase, we let n'_h be the maximum noise in any ciphertext in W_1 at level h of the search tree. We have that W_1 at level h is computed as the sum of $(k+1)^2$ sums of products of noise level $n'_{h-1} \cdot (n^{3/2} \cdot t^2)^2$ (i.e., each sum is a product of two ciphertexts of noise level n'_{h-1}). Thus $n'_{\log_2|Y|-1} = ((k+1) \cdot (n^{3/2} \cdot t^2)^{2\log_2|Y|} \cdot n'_0$. Taking $n'_0 = n_M$, yields $n'_{\log_2|Y|-1} < (k+1)^{2\log_2|Y|} \cdot (4 \cdot n^{3/2})^{2\log_2|Y|} \cdot 2 \cdot (4 \cdot (4k)^{1-c} \cdot n^{3/2})^{(4k)^c} \cdot 2 \cdot (4 \cdot n^{1/2} \cdot B_{err}) < \frac{q}{2}$, yielding the result. □

Lemma 6. Closest-Matching *achieves security against statically chosen semi-honest adversaries if the* Decision-$\mathsf{LWE}_{2n,q,\chi}$ *and* Decision-$\mathsf{SPR}_{2n,q,\chi}$ *assumptions hold.*

Proof. Simulator $\mathcal{S}$ proceeds as follows. Generate $(4k)^{1/2} \cdot 2^{(4k)^{1/2}}$ random ciphertexts and forwards them to P_2. On receipt of the ciphertexts by P_2, $\mathcal{S}$ generates $O(1)$ random ciphertexts and forwards them to P_1. By a hybrid argument for any PPT adversary $\mathcal{A}$ with advantage ϵ distinguishing the real protocol from the simulation, we can construct a PPT adversary $\mathcal{A}'$ with advantage at least $\frac{\epsilon}{(4k)^{1/2} \cdot 2^{(4k)^{1/2}} + O(1)}$ in breaking either the Decision-$\mathsf{LWE}_{2n,q,\chi}$ assumption or the Decision-$\mathsf{SPR}_{2n,q,\chi}$ assumption. Since k is a constant, the claim follows. □

7 Performance

In this section we evaluate the theoretical resource costs of the protocols described in the previous three sections. Since key generation and decryption are one-time procedures and substantially less expensive than homomorphic evaluation, we omit them from our analysis. We also omit composition by CRT from our analysis because it can performed independently of encryption. We note that Lemmas 1, 3 and 5 enable selection of suitable parameters to guarantee correctness of the protocols with all but negligible probability for corresponding input sizes, while our analysis assumes fixed sets of such parameters.

Protocol	Encryption cost	Evaluation cost	Transfer size
Set Intersection with labelling I	$\|X\|\alpha$	$\frac{\|X\|\|Y\|(\beta+\gamma)}{n}$	$(\|X\| + \frac{\|Y\|}{n})\sigma$
Set Intersection with labelling II	$\frac{4k\|X\|\alpha}{n}$	$8k\|Y\|\log_2(k\|Y\|)(2\beta+\gamma)$	$8k\|Y\|\sigma$
Fuzzy Matching with labelling	$\frac{16k^2\|X\|\alpha}{n}$	$\frac{4k\|X\|\|Y\|(4k(\beta+\delta)+\gamma))}{n}$	$\frac{4k\|X\|\|Y\|\sigma}{n}$
Closest Matching	$(4k)^c \cdot 2^{(4k)^{1-c}}\alpha$	$k^2(\|Y\| + 16k)(\beta+\delta)$	$(4k)^c \cdot 2^{(4k)^{1-c}}\sigma$

Table 1: Time and space costs for the three protocols for fixed parameters t, n and q. Labels assumed to be $\theta(k)$ bits.

7.1 Comparison of Protocols

Table 1 shows the time measured in aggregate atomic homomorphic operations and space measured in number of ciphertexts. For encryption parameters t, n and q let $\alpha_{(t,n,q)}, \beta_{(t,n,q)}, \gamma_{(t,n,q)}$ and $\delta_{(t,n,q)}$ be the time taken for encryption, ciphertext addition, ciphertext multiplication by a plaintext polynomial and ciphertext multiplication without relinearization respectively. Let $\sigma_{(t,n,q)}$ be the associated ciphertext size.

7.2 Empirical Results

We have implemented the basic functionality corresponding to each of the three protocols in the recently published homomorphic cryptography library SEAL [13]. The machine was a desktop running Ubuntu with 16Gb RAM and an Intel Core i7-4790 @3.60 GHz processor. The corresponding time and space consumption for some input sizes is shown in Table 2. For comparison, we present the times yielded by the pattern match approach of [14] applied to plain private set intersection. Specifically we treat every client k-mer as a position-independent pattern to be matched against each k-mer in the server set assuming EC-ElGamal encryption is used to encrypt each nucleotide.

Protocol	k	$\|X\|$	$\|Y\|$	t	n	q	Encryption (Minutes)	Evaluation (Hours)	Transfer (Gb)
Set Intersection with labelling I	32	2^{12}	2^{12}	$\approx 2^{64}$	16384	$\approx 2^{768}$	61	<1	7
Set Intersection with labelling II	32	2^{12}	2^{12}	$\approx 2^{64}$	16384	$\approx 2^{768}$	<1	9	29
Set Intersection [14]	32	2^{12}	2^{12}	-	-	-	7	96	1
Fuzzy matching with labelling	32	2^{8}	2^{8}	163841	8192	$\approx 2^{384}$	2	15	1
Closest Matching	32	-	2^{8}	2	16384	$\approx 2^{768}$	9	136	1

Table 2: Estimated time and space costs for the three protocols.

References

[1] Malin, B., Sweeney, L.: Determining the identifiability of DNA database entries. In: AMIA Annual Symposium Proceedings. (2000) 537–541

[2] Gymrek, M., McGuire, A.L., Golan, D., Halpern, E., Ehrlich, Y.: Identifying personal genomes by surname inference. Science **339** (2013) 321–324

[3] Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., Muehling, J., Pearson, J.V., Stephan, D.A., Nelson, S.F., Craig, D.W.: Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. PLoS Genetics **4** (2008)

[4] Wang, R., Li, Y.F., Wang, X., Tang, H., Zhou, X.: Learning your identity and disease from research papers: Information leaks in genome wide association study. In: Proceedings of the 16th ACM Conference on Computer and Communications Security. CCS '09, New York, NY, USA, ACM (2009) 534–544

[5] Humbert, M., Ayday, E., Hubaux, J.P., Telenti, A.: Addressing the concerns of the lacks family: Quantification of kin genomic privacy. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. CCS '13, New York, NY, USA, ACM (2013) 1141–1152

[6] Nyholt, D.R., Yu, C.E., Visscher, P.M.: On Jim Watson's APOE status: Genetic information is hard to hide. European Journal of Human Genetics **17** (2009) 147–149

[7] Bedo, J., Goudey, B., Wazny, J., Zhou, Z.: Information theoretic alignment free variant calling. PeerJ Computer Science **2** (2016) e71

[8] Steinlein, O.K.: Genes and mutations in idiopathic epilepsy. American Journal of Medical Genetics **106** (2001) 139–145

[9] Köksal, M.O., Beka, H., Demirci, M., Kadioglu, A., Agacfidan, A., Akgül, B.: Prevalence and genotyping of chlamydia trachomatis in symptomatic male patients from istanbul, turkey. SpringerPlus **5** (2016) 1706

[10] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In Cachin, C., Camenisch, J.L., eds.: Advances in Cryptology - EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg (2004) 1–19

[11] Kissner, L., Song, D.: Privacy-preserving set operations. In: Proceedings of the 25th Annual International Conference on Advances in Cryptology. CRYPTO'05, Berlin, Heidelberg, Springer-Verlag (2005) 241–257

[12] Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In Stam, M., ed.: Cryptography and Coding: 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg (2013) 45–64

[13] Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Manual for using homomorphic encryption for bioinformatics. Technical Report MSR-TR-2015-87 (2015)

[14] De Cristofaro, E., Faber, S., Tsudik, G.: Secure genomic testing with size- and position-hiding private substring matching. In: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society. WPES '13, New York, NY, USA, ACM (2013) 107–118

[15] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. ITCS '12, New York, NY, USA, ACM (2012) 309–325

[16] Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In Safavi-Naini, R., Canetti, R., eds.: Advances in Cryptology – CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg (2012) 850–867

[17] López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing. STOC '12, New York, NY, USA, ACM (2012) 1219–1234

[18] Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology. EUROCRYPT'11, Berlin, Heidelberg, Springer-Verlag (2011) 27–47

[19] Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012) `http://eprint.iacr.org/`.

[20] Rindal, P., Rosulek, M.: Improved private set intersection against malicious adversaries. Cryptology ePrint Archive, Report 2016/746 (2016) `http://eprint.iacr.org/2016/746`.

[21] Pinkas, B., Schneider, T., Segev, G., Zohner, M.: Phasing: Private set intersection using permutation-based hashing. Cryptology ePrint Archive, Report 2015/634 (2015) `http://eprint.iacr.org/2015/634`.

[22] Rabin, M.O.: How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University (1981)

[23] Kim, M., Lauter, K.: Private genome analysis through homomorphic encryption. BMC Med Inform Decis Mak **15** (2015)

[24] Cheon, J.H., Kim, M., Lauter, K.: Homomorphic computation of edit distance. In Brenner, M., Christin, N., Johnson, B., Rohloff, K., eds.: Financial Cryptography and Data Security: FC 2015 International Workshops, BITCOIN, WAHC, and Wearable, San Juan, Puerto Rico, January 30, 2015, Revised Selected Papers. Springer Berlin Heidelberg, Berlin, Heidelberg (2015) 194–212

[25] Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Secure pattern matching using somewhat homomorphic encryption. In: Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop. CCSW '13, New York, NY, USA, ACM (2013) 65–76

[26] Ayday, E., Raisaro, J.L., McLaren, P.J., Fellay, J., Hubaux, J.P.: Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data. In: Proceedings of the 2013 USENIX Conference on Safety, Security, Privacy and Interoperability of Health Information Technologies. HealthTech'13, Berkeley, CA, USA, USENIX Association (2013) 1–1

[27] Danezis, G., De Cristofaro, E.: Fast and private genomic testing for disease susceptibility. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society. WPES '14, New York, NY, USA, ACM (2014) 31–34

[28] Bruekers, F., Katzenbeisser, S., Kursawe, K., Tuyls, P.: Privacy-preserving matching of dna profiles. Cryptology ePrint Archive, Report 2008/203 (2008) `http://eprint.iacr.org/2008/203`.

[29] Zhao, Y., Chow, S.S.: Are you the one to share? secret transfer with access structure. Cryptology ePrint Archive, Report 2015/929 (2015) `http://eprint.iacr.org/2015/929`.

[30] Ye, Q., Steinfeld, R., Pieprzyk, J., Wang, H.: Efficient fuzzy matching and intersection on private datasets. In Lee, D., Hong, S., eds.: Information, Security and Cryptology – ICISC 2009: 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers, Berlin, Heidelberg, Springer Berlin Heidelberg (2010) 211–228

[31] Lepoint, T., Naehrig, M.: A comparison of the homomorphic encryption schemes FV and YASHE. In Pointcheval, D., Vergnaud, D., eds.: Progress in Cryptology – AFRICACRYPT 2014: 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings. Springer International Publishing, Cham (2014) 318–335

[32] Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In Garay, J.A., Gennaro, R., eds.: Advances in Cryptology – CRYPTO 2014: 34th Annual

Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I. Springer Berlin Heidelberg, Berlin, Heidelberg (2014) 297–314

[33] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing. STOC '05, New York, NY, USA, ACM (2005) 84–93

[34] Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols. Information Security and Cryptography. Springer-Verlag Berlin Heidelberg (2010)

[35] Goh, E.J., Golle, P.: Event driven private counters. In Patrick, A.S., Yung, M., eds.: Financial Cryptography and Data Security: 9th International Conference, FC 2005, Roseau, The Commonwealth Of Dominica, February 28 – March 3, 2005. Revised Papers. Springer Berlin Heidelberg, Berlin, Heidelberg (2005) 313–327

[36] Benaloh, J., Moran, T., Naish, L., Ramchen, K., Teague, V.: Shuffle-sum: Coercion-resistant verifiable tallying for STV voting. In: IEEE Transactions on Information Forensics and Security. Volume 4., IEEE (2009) 685–698

[37] Storer, J.A.: An Introduction to Data Structures and Algorithms. Birkhäuser Boston, Boston, MA (2002)

[38] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Proceedings on Advances in cryptology—CRYPTO '86, London, UK, UK, Springer-Verlag (1987) 186–194

A Set Intersection with Quasilinear Complexity

In this section, we describe a private set intersection protocol from homomorphic encryption with quasilinear complexity, based upon the multi-round protocol by Kissner and Song [11]. Let s be the maximum of $|X|$ and $|Y|$. In this protocol the client and server proceed by constructing polynomials α and β which vanish on their respective input sets, the coefficients of these polynomials are encrypted under a threshold cryptosystem with an additive homomorphism. Additionally the client and server generate random polynomials μ and ν of degree s. Using the additive homomorphism, the client and server compute an encryption of the polynomial $q = \alpha \cdot \mu + \beta \cdot \nu$. Both players then decrypt to recover q. Lemma 2 [11] establishes that $q = \gcd(\alpha, \beta) \cdot u$ where u is a random polynomial of degree m with overwhelming probability. Thus both parties learn the zeroes in common of their polynomials, i.e., the intersection of their sets – and nothing else.

Starting with this framework we construct a non-interactive private set intersection scheme with quasilinear complexity as follows. Firstly the client encodes the coefficients of the polynomial α as individual ciphertexts, the list is padded with dummy encryptions of zero to form an array of length $2s$. They send this list to the server. The server then chooses random polynomials μ and ν of degree s and computes the product polynomials

$\alpha \cdot \mu$ and $\beta \cdot \nu$. Crucially this step can be performed in sub-quadratic time, for example using a fast number-theoretic transform. These products, still in coefficient form are then homomorphically added, yielding their sum $\alpha \cdot \mu + \beta \cdot \nu$, which is then returned to the client.

Having adapted [11] to the non-interactive case, our solution to the problem of transferring labels uses a bit-by-bit approach.[2]

Let k' be the maximum length of diagnostic labels. The client uses their private set X to generated the expanded set $X^* = \{x\|i\|0, x\|i\|1 : x \in X, i \in [k']\}$. The server on the other hand generates the set $Y^* = \{y\|i\|\ell(y)_i : y \in Y, i \in [k']\}$. Given the intersection of the sets X^* and Y^*, the client can reconstruct the k-mers in common and their corresponding labels, by extracting the latter one bit at a time. The overhead to the original protocol is only a multiplicative factor k'.

Noting that all operations described so far are linear, we may efficiently batch this protocol as follows. The client splits their expanded set X^* into n equal size sets, constructing interpolating polynomials for each sets. These polynomials, say $\alpha^{(1)}, \dots, \alpha^{(n)}$ are stored in batch co-efficient representation in $\frac{|X^*|}{n}$ ciphertexts. The server generates corresponding random polynomials $\mu^{(1)}, \dots, \mu^{(n)}$ and $\nu^{(1)}, \dots, \nu^{(n)}$. The server then computes $\alpha^{(j)}\mu^{(j)} + \beta\nu^{(j)}$ for $j = 1, \dots, n$. In this way the client obtains $(\gcd(\alpha^{(j)}, \beta))_{j=1}^n$, thus deducing the total intersection as a union of n smaller intersections.

Protocol Description – Scenario 1

The client and server begin by constructing their expanded sets X^* and Y^*. Let $I = \frac{|X^*|}{n}$. The server splits X^* into n batches, constructing the polynomials $\alpha^{(j)} = \sum_{i=0}^{I-1} a_i^{(j)} x^i$ for $j = 1, \dots, n$. These polynomials are encrypted in batch form in I ciphertexts. The server then constructs β for their expanded set $|Y^*|$, as well as random polynomials $\mu^{(j)}$ and $\nu^{(j)}$ of degree $s = \max\{\frac{|X^*|}{n}, |Y^*|\}$. The client sends the encrypted coefficients of their polynomials to the server. The server uses a fast number theoretic transform to homomorphically compute $\alpha^{(j)}\mu^{(j)}$, for $j = 1, \dots, n$ in a quasilinear number of homomorphic SIMD operations. The server also encrypts the individual coefficients of the polynomials $(\beta\nu^{(j)})_{j=1}^n$ as an array of ciphertexts. In linear time they batch homomorphically add the polynomials $\alpha^{(j)}\mu^{(j)}$ and $\beta\nu^{(j)}$, for $j = 1, \dots, n$ together and return the resulting encrypted coefficients of the polynomials to the client.

For the fast polynomial multiplication step, we use a modified form of the in-place, bottom-up version of the Fast Fourier Transform described on page 436, Chapter 12 [37]. In the bit-reversal phase of the modified algorithm, we will use i^R to denote the integer corresponding to the reversed bit representation of index $i \in [2s]$.

Lemma 7. *Let* $k' = \max_{y \in Y} |\ell(y)|$. Set-Intersection-II *correctly computes set intersection with labelling if* $4 \cdot 2^{(2k + \log_2 k' + 1)(2\log_2(k|X|)+5)} \cdot n^{\log_2(k|X|)+2} \cdot B_{err} < q$ *except with* $\mathrm{negl}(\lambda)$ *probability.*

[2] An efficient implementation should use a word size greater than one bit. We stick with a one-bit word size in the following description for conceptual clarity.

procedure SET-INTERSECTION-II$(X, Y, \ell(\cdot))$

  $\mathbf{P_1}, \mathbf{P_2}$: $k' = \max_{y \in Y} |\ell(y)|$, $s \leftarrow \max\{\frac{|X^*|}{n}, |Y^*|\}$,

    $t \leftarrow p_{\geq 2^{2k+\log_2 k'+1}} : t \equiv 1 \bmod 2s$

  $\mathbf{P_1}$:

    Let $X^* = \cup_{x \in X, i \in [k'], b \in \{0,1\}} x \| i \| b$

    Enumerate X^* as $\{x_1, \ldots, x_{|X^*|}\}$

    for $j \leftarrow 1$ to n **do**

      $I \leftarrow \frac{|X^*|}{n}$

      Construct $\alpha^{(j)} = \sum_{i=0}^{I-1} a_i^{(j)} x^i$ such that

      $\alpha^{(j)}(x_{(j-1)I+l}) = 0$ for $l = 1, \ldots, I$.

    end for

    $A \leftarrow (\mathsf{Enc}(\mathsf{CRT}(a_0^{(1)}, \ldots, a_0^{(n)}), \mathsf{pk}), \ldots,$

    $\mathsf{Enc}(\mathsf{CRT}(a_{I-1}^{(1)}, \ldots, a_{I-1}^{(n)})), \mathsf{pk}),$

    $\underbrace{\mathsf{Enc}^*(0, \mathsf{pk}), \ldots, \mathsf{Enc}^*(0, \mathsf{pk})}_{2s-I})$

    Send A.

  $\mathbf{P_2}$:

    Let $Y^* = \cup_{y \in Y, i \in [k']} y \| i \| \ell(y)_i$

    Enumerate Y^* as $\{y_1, \ldots, y_{|Y^*|}\}$

    Choose $w \in \mathbb{Z}_t^* : \mathrm{ord}_t(w) = 2s$

    Construct $\beta = \sum_{i=0}^{|Y^*|-1} b_i x^i$ such that

    $\beta(y_i) = 0$ for $i = 1, \ldots, |Y^*|$.

    for $j \leftarrow 1$ to n **do**

      Let $\mu^{(j)}(x) = \sum_{i=0}^{s} m_i^{(j)} x^i$ where

      $m_i^{(j)} \in_R \mathbb{Z}_t$ for $i = 1, \ldots, s$ and $j = 1, \ldots, n$

      Let $\nu^{(j)}(x) = \sum_{i=0}^{s} n_i^{(j)} x^i$ where

      $n_i^{(j)} \in_R \mathbb{Z}_t$ for $i = 1, \ldots, s$ and $j = 1, \ldots, n$

      Compute $\upsilon^{(j)}(x) = \beta(x)\nu^{(j)}(x) =$

      $\sum_{i=0}^{2s-1} u_i^{(j)} x^i$, for $j = 1, \ldots, n$.

    end for

    $A' \leftarrow$ HOM-TRANSFORM$(A, 2s, \mathsf{false})$

    for $i \leftarrow 1$ to $2s$ **do**

      $T'_i \leftarrow A'_i \ltimes \mathsf{CRT}(\mu^{(1)}(w^{i-1}), \ldots, \mu^{(n)}(w^{i-1}))$

    end for

    $T \leftarrow$ HOM-TRANSFORM$(T', 2s, \mathsf{true})$

    $U \leftarrow (\mathsf{Enc}(\mathsf{CRT}(u_0^{(1)}, \ldots, u_0^{(n)}), \mathsf{pk}), \ldots,$

    $\mathsf{Enc}(\mathsf{CRT}(u_{2s-1}^{(1)}, \ldots, u_{2s-1}^{(n)}), \mathsf{pk}))$

    for $i \leftarrow 1$ to $2s$ **do**

      $\mathsf{ct}_i \leftarrow T_i + U_i$

    end for

    Send $(\mathsf{ct}_1, \ldots, \mathsf{ct}_{2s})$.

end procedure

```
procedure Hom-Transform(A, n_A, inv)
    for i ← 0 to n_A − 1 do
        if i < i^R then
            A_{i+1} ↔ A_{i^R+1}
        end if
    end for
    if inv then
        z ← w^{n_A−1}
    else
        z ← w
    end if
    for h ← 0 to log_2(n_A) − 1 do
        for i ← 0 to n_A − 2^{h+1} step 2^{h+1} do
            for j ← 1 to 2^{h+1} do
                ct_(⋉) ← A_{i+j+2^h} ⋉
                CRT((z^{(n_A/2^{h+1})})^{j−1}, . . . , (z^{(n_A/2^{h+1})})^{j−1})
                A_{i+j+2^h} ← A_{i+j} − ct_(⋉)
                A_{i+j} ← A_{i+j} + ct_(⋉)
            end for
        end for
    end for
    if inv then
        for i ← 1 to n_A do
            A_i ← A_i ⋉ CRT(n_A^{−1}, . . . , n_A^{−1})
        end for
    end if
end procedure
```

Proof. We consider the noise contribution to the output ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_{2s})$ corresponding to each of the calls to the homomorphic Fast Fourier Transform (FFT) procedure. In the first call the FFT circuit is batch evaluated homomorphically on the array A, which contains input ciphertexts of noise level n_{fresh}. The ciphertexts produced at the output of this circuit, are then batch multiplied by plaintext evaluations of the polynomials $\mu^{(j)}$ at the $2s$ powers of the $2s^{th}$ root of unity w, yielding the batch transforms of $\alpha^{(j)}\mu^{(j)}$ as an array of ciphertexts T'. In the second stage the inverse FFT is evaluated homomorphically on T', yielding T. Clearly the noise contribution from this stage is identical to the that of the first stage. Let $L = \log_2(2s)$ and let $n_1, \ldots, n_{L+1}, n'_1, \ldots, n'_{L+1}$ be the maximum noise level of the ciphertexts at each level of the forward and inverse homomorphic FFT stages, respectively. We have that $n_{l+1} \leq n_l + n_l \cdot z_{l,j} \cdot n^{1/2} + t$ where $z_{l,j} = w^{\frac{2s(j-1)}{2^l}}$. Since $z_{l,j}$ is simply a scalar in $\mathbb{Z}_t$, it holds that, regardless of the values of l and j, $z_{l,j} \leq t$. Then $n_{L+1} \leq n_1(1+t)^L n^{L/2} + t\sum_{l=1}^{L-1}(1+t)^l n^{L/2} \leq n_1(1+t)^L n^{L/2} + t(1+t)^L n^{L/2}$. Thus $n_{L+1} \leq (n_1 + t)(1+t)^L n^{L/2}$. Also $n'_1 \leq (1+t)n_{L+1}n^{1/2}$, corresponding to a simple multiplication by the plaintexts $(\mu(w^i))_{i=1}^{2s}$. Finally, by symmetry, $n'_{L+1} \leq (n'_1 + t)(1+t)^L n^{L/2}$. It follows that $n'_{L+1} \leq (1+t)^{2L+1} n^L (n_1 + t)$. Now, since $n_1 = n_{\text{fresh}}$, we have:

$$\begin{aligned} n'_{L+1} &\leq t(1+t)^{2L+1} n^L (1 + 2n^{1/2} B_{err}) \\ &\leq 2(1+t)^{2L+2} n^{L+(1/2)} B_{err} \\ &\leq 2t^{2L+3} n^{L+1} B_{err} \\ &\leq 2 \cdot 2^{(2k+\log_2 k'+1)(2L+3)} \cdot n^{L+1} \cdot B_{err} \\ &< \frac{q}{2} \end{aligned}$$

completing the proof. □

Lemma 8. Set-Intersection-II *achieves security against statically chosen semi-honest adversaries if the* Decision-$\mathsf{LWE}_{2n,q,\chi}$ *and* Decision-$\mathsf{SPR}_{2n,q,\chi}$ *assumptions hold.*

Proof. Simulator $\mathcal{S}$ proceeds as follows. Generate $\frac{2k'|X|}{n}$ random ciphertexts and forward them to P_2. On receipt of the ciphertexts by P_2, $\mathcal{S}$ generates $4k' \max\{|X|, |Y|\}$ random ciphertexts and forwards them to P_1. By a hybrid argument for any PPT adversary $\mathcal{A}$ with advantage ϵ distinguishing the real protocol from the simulation, we can construct a PPT adversary $\mathcal{A}'$ with advantage at least $\frac{\epsilon}{\frac{2k'|X|}{n} + 4k' \max\{|X|,|Y|\}}$ in breaking either the Decision-$\mathsf{LWE}_{2n,q,\chi}$ assumption or the Decision-$\mathsf{SPR}_{2n,q,\chi}$ assumption. Since $|X|$, $|Y|$ and k' are constants, the claim follows. □

B Protocols Secure against Malicious Adversaries

B.1 Set Intersection with Labelling

In this section, we describe a protocol for set intersection with labelling secure against malicious adversaries, following a similar procedure to the modifications to the semi-honest protocols in Section 4, [10] described in Section 5.3 of the same work.

Let N be a security parameter. The client generates N copies of their input for use in a cut-and-choose protocol with the server. Each copy is constructed by the client first mapping their input to pseudonyms via a random oracle applied to a seed concatenated with the input k-mer. Each copy of the input set utilises a different seed. To achieve the challenge phase in our offline/online setting, we exploit the Fiat-Shamir heuristic [38]. Specifically, the client applies a strong hash function to the entire set of copies to produce the subset that will be opened. Along the with openings of this subset of encrypted pseudonyms, the client also sends the corresponding seeds. Together, this enables the server to verify that the majority of the input sets are correctly constructed, with overwhelming probability. To achieve security against a malicious server, the server transfers in place of a matching k-mer a short seed to third hash function. The output of this hash function is used to de-randomize the rest of the computation. Let t be a random λ-bit value and $H(t) = u' \| u''$ and let $(\mathsf{Enc}(a_i))_{i=0}^{|X|-1}$ be an encryption of client polynomial $P(\cdot)$. For k-mer y, the server homomorphically computes $\mathsf{Enc}(u' \cdot P(y) + t)$ as well as the value $z = H(u'', y)$. In the event that y is also a client k-mer, the seed t is recoverable by decryption, from which y is determined to be matching k-mer by brute-force testing of H on every k-mer in the client set against z. Security against a malicious server thus follows from the improbability that random oracle H produces the same output on two different $2k$-bit suffixes. We require the following notation.

Let $H_1 : \{0,1\}^{2k+\lambda} \rightarrow \{0,1\}^{2k+\lambda}, H_2 : \{0,1\}^{N\cdot|X|\cdot|\mathsf{ct}|} \rightarrow \{0,1\}^N, H_3 : \{0,1\}^{\lambda+\log_2|Y|} \rightarrow \{0,1\}^{2k+2\lambda}, H_4 : \{0,1\}^{2k+\lambda} \rightarrow \{0,1\}^{2k+\lambda}$ be random oracles. Let $\mathcal{F}_{\mathsf{L\text{-}Set\text{-}Int}}$ be the functionality corresponding to Task 1, as it is defined in Section 3. Let $\mathcal{F}^i_{\mathsf{RO}}$ be the functionality corresponding to the random oracle H_i, for $1 \leq i \leq 4$.

Security of our protocol follows from Theorem 9, assuming the Decision-$\mathsf{LWE}_{2n,q,\chi}$ and Decision-$\mathsf{SPR}_{2n,q,\chi}$ assumptions hold.

Theorem 9. *Assuming that H_1-H_4 are random oracles and that $\mathsf{Enc}(\cdot,\cdot)$ is a message encryption function of a semantically secure cryptosystem* Set-Intersection-I* *achieves security against statically chosen malicious adversaries.*

Proof. We analyse security of the protocol in the hybrid world where a trusted third party computes the functionalities $\mathcal{F}^1_{\mathsf{RO}}$-$\mathcal{F}^4_{\mathsf{RO}}$ for both client and server.

In the case of malicious $\mathbf{P_1}$, simulator $\mathcal{B}$ in the ideal world proceeds as follows.

1. In each call to the hash function H_1, $\mathcal{B}$ learns the input of $\mathcal{A}$, namely (s_ι, x_i) for $i \in [|X|], \iota \in [N]$. It sends $\mathcal{A}$ the value $h^1_{(i,\iota)} = \mathcal{F}^1_{\mathsf{RO}}(s_\iota, x_i)$ and stores $(s_\iota, x_i, h_{(i,\iota)})$ locally.
2. Let $X^{(\iota)} = (h^1_{(i,\iota)})_{i=1}^{|X|}$. $\mathcal{B}$ sends $X^{(\iota)}$ to $\mathcal{F}_{\mathsf{L\text{-}Set\text{-}Int}}$ and receives $L^{(\iota)} = (l_i^{(\iota)})_{i=1}^{|X\cap Y|}$.
3. In the call to hash function H_2, $\mathcal{B}$ learns ciphertext set S, sends $\mathcal{A}$ the value $J = \mathcal{F}^2_{\mathsf{RO}}(S)$ and stores S locally.
4. $\mathcal{B}$ receives from $\mathcal{A}$ the set $(s_\iota)_{\iota\in[N]\setminus J'}$ and openings of $(\mathsf{ct}_i^{(\iota)})_{i\in[|X|],\iota\in J'}$ for some set $J' \subset [N]$. It verifies that $J' = J$ and that the received seeds are consistent with

procedure SET-INTERSECTION-I*$(X, Y, \ell(\cdot))$

P_1, P_2 : $I \leftarrow \frac{|Y|}{n}, t \leftarrow p_{\geq 2^{2k+\lambda}}$

P_1 :

Enumerate X as $\{x_1, \ldots, x_{|X|}\}$

Choose $s_0, \ldots, s_{N-1} \in_R \{0,1\}^\lambda$

for $\iota \leftarrow 0$ to $N-1$ **do**

Construct $P^{(\iota)}(\cdot) = \sum_{i=0}^{|X|-1} a_i^{(\iota)} x^i$:

$P^{(\iota)}(H_1(s_\iota, x_i)) = 0$ for $i = 1, \ldots, |X|$.

for $i \leftarrow 1$ to $|X|$ **do**

$\mathsf{ct'}_i^{(\iota)} \leftarrow \mathsf{Enc}(\mathsf{CRT}(a_{i-1}^{(\iota)}, \ldots, a_{i-1}^{(\iota)}), \mathsf{pk})$

end for

end for

Send $(\mathsf{ct'}_1^{(\iota)}, \ldots, \mathsf{ct'}_{|X|}^{(\iota)})_{\iota=0}^{N-1}$.

$J \leftarrow H_2((\mathsf{ct'}_1^{(\iota)}, \ldots, \mathsf{ct'}_{|X|}^{(\iota)})_{\iota=0}^{N-1})$.

Send $(s_\iota)_{\iota \in [N] \setminus J}$.

Send openings of $(\mathsf{ct'}_i^{(\iota)})_{i \in [|X|],\ \iota \in J}$.

P_2 :

Enumerate Y as $\{y_1, \ldots, y_{|Y|}\}$

Verify openings and correctness of J,

output $\perp$ if verification fails.

for $\iota \in [N] \setminus J$ **do**

Pick $t^{(\iota)} \in_R \{0,1\}^\lambda$

for $i \leftarrow 1$ to I **do**

for $j \leftarrow 1$ to $|X|$ **do**

$p_i^{(\iota,j)} \leftarrow \mathsf{CRT}(H_1(s_\iota, y_{(i-1)n+1})^{j-1},$

$\ldots, H_1(s_\iota, y_{in})^{j-1})$

end for

Write $H_3(t^{(\iota)} \| i \| 1) = {u'_{i1}}^{(\iota)} \| {u''_{i1}}^{(\iota)}, \ldots,$

$H_3(t^{(\iota)} \| i \| n) = {u'_{in}}^{(\iota)} \| {u''_{in}}^{(\iota)}$

$p_i^{(\iota)} \leftarrow \mathsf{CRT}({u'_{i1}}^{(\iota)}, \ldots, {u'_{in}}^{(\iota)})$

$z_i^{(\iota)} \leftarrow \mathsf{CRT}(H_4({u''_{i1}}^{(\iota)}, y_{(i-1)n+1}), \ldots,$

$H_4({u''_{in}}^{(\iota)}, y_{in}))$

$q_i^{(\iota)} \leftarrow \mathsf{CRT}(\ell(y_{(i-1)n+1}) \| t^{(\iota)},$

$\ldots, \ell(y_{in}) \| t^{(\iota)})$

$\mathsf{ct}_i^{(\iota)} \leftarrow (\sum_{j=1}^{|X|} \mathsf{ct'}_i^{(\iota)} \ltimes p_i^{(\iota,j)}) \ltimes p_i^{(\iota)} + q_i^{(\iota)}$

end for

end for

Send $(\mathsf{ct}_1^{(\iota)}, \ldots, \mathsf{ct}_I^{(\iota)})_{\iota \in [N] \setminus J}$ and

$(z_1^{(\iota)}, \ldots, z_I^{(\iota)})_{\iota \in [N] \setminus J}$.

P_1 :

Decrypt $(\mathsf{ct}_i^{(\iota)})_{i \in [I], \iota \in [N] \setminus J}$, and verify correctness

w.r.t $(z_i^{(\iota)})_{i \in [I], \iota \in [N] \setminus J}$.

end procedure

those received in Step 1 and that the opened ciphertexts are consistent with $S|_J$. Output $\perp$ if any of these checks fail.

5. Let $a = \frac{n \cdot |X \cap Y|}{|Y|}$. Perform the following for $\iota \in [N]$ and $i \in [I]$. Construct $q_i^{(\iota)}$ as a batch encryption of a values from $L^{(\iota)}$ and $n - a$ random $2k + \lambda$-bit strings. Generate $p_i^{(\iota,j)}$ and $p_i^{(\iota)}$ as the honest server would. Compute $\mathsf{ct}_i^{(\iota)}$ using $p_i^{(\iota,j)}, p_i^{(\iota)}$ and $q_i^{(\iota)}$. Compute $z_i^{(\iota)}$ honestly. Send $(\mathsf{ct}_i^{(\iota)})_{i \in [|X|], \iota \in [N] \setminus J}$ and $(z_i^{(\iota)})_{i \in [|X|], i \in [N] \setminus J}$ to $\mathcal{A}$.
6. Output whatever $\mathcal{A}$ outputs.

That the above simulation against a malicious client in the ideal world is indistinguishable from the real-world execution in the hybrid model, follows from the fact that the only place where the simulation differs from the real world is in Step 5. Here the input ciphertext $\mathsf{ct}_i^{(\iota)}$, where $i \in [I]$ and $\iota \in [N]$, is computed directly using $q_i^{(\iota)}$ constructed from the output of the trusted party computing $\mathcal{F}_{\mathsf{L\text{-}Set\text{-}Int}}$ in Step 2. This is possible without $\mathcal{A}$ noticing, because in the $(\mathcal{F}^3_{\mathsf{RO}}, \mathcal{F}^4_{\mathsf{RO}})$-hybrid model, the output of H_3 and H_4 is indistinguishable from the uniform distribution on $\{0,1\}^{2k+2\lambda}$ and $\{0,1\}^{2k+\lambda}$, respectively.

In the case of malicious $\mathbf{P_2}$, simulator $\mathcal{B}$ proceeds as follows.

1. Generate random strings $h^1_{(i,\iota,l)}$ in $\{0,1\}^{2\lambda+k}$ for each $(i, \iota, l) \in [|X|] \times [N] \times [n]$. Compute $(\mathsf{ct'}_i^{(\iota)})_{i \in [|X|], \iota \in [N]}$ using inputs $h_{(i,\iota,l)}$ and compute J as the honest client would. Send $(s_\iota)_{\iota \in [N] \setminus J}$ and openings of $(\mathsf{ct'}_i^{(\iota)})_{i \in [|X|], \iota \in J}$ to $\mathcal{A}$.
2. In each call to hash function H_1, $\mathcal{B}$ replaces the output of $\mathcal{F}^1_{\mathsf{RO}}$ with the string $h^1_{(i,\iota,l)}$ for input $(s_\iota, y_{(i-1)n+l})$ iff $y_{(i-1)n+l} = x_{(i-1)n+l}$.
3. In each call to hash function H_3, $\mathcal{B}$ learns $(t^{(\iota)} \| i \| l)$ for $i \in [I], \iota \in [N] \setminus J, l \in [n]$. It sends $\mathcal{A}$ the value ${u_{il}}^{(\iota)} = \mathcal{F}^3_{\mathsf{RO}}(t^{(\iota)} \| i \| l)$ and stores $(t^{(\iota)}, ({u_{il}}^{(\iota)})_{l \in [n]})$ locally.
4. In each call to hash function H_4, $\mathcal{B}$ learns the values $({u''_{il}}^{(\iota)}, y_{(i-1)n+l})$ for $i \in [I], \iota \in [N] \setminus J, l \in [n]$. It sends $\mathcal{A}$ the values $h^4_{(i,\iota,l)} = \mathcal{F}^4_{\mathsf{RO}}({u''_{il}}^{(\iota)}, y_{(i-1)n+l}) : i \in [I], \iota \in [N] \setminus J, l \in [n]$. It stores $(({u''_{i1}}^{(\iota)}, y_{(i-1)n+1}), \ldots, ({u''_{in}}^{(\iota)}, y_{in}))$ and $(h^4_{(i,\iota,l)})_{i \in [I], \iota \in [N] \setminus J, l \in [n]}$ locally.
5. Let $Y^{(\iota)} = \{h^4_{(i,\iota,l)}\}_{i \in [I], l \in [n]}$. For $\iota \in [N]$ it forwards the set of inputs $Y^{(\iota)}$ to $\mathcal{F}_{\mathsf{L\text{-}Set\text{-}Int}}$ and receives $L^{(\iota)} = (\ell_i^{(\iota)})_{i=1}^{|X \cap Y|}$.
6. $\mathcal{B}$ receives $(\mathsf{ct}_i^{(\iota)})_{i \in [I], \iota \in [N] \setminus J}$ and $(z_i^{(\iota)})_{i \in [I], \iota \in [N] \setminus J}$. It verifies the ι^{th} batch of ciphertexts corresponds to label set $L^{(\iota)}$ and outputs 1 if $(z_i^{(\iota)})_{i \in [I], \iota \in [N] \setminus J}$ are formed correctly.

We argue the real world and ideal world executions are indistinguishable in $\mathcal{F}^1_{\mathsf{RO}}$-hybrid model as follows. The only place where the simulation differs from the protocol is in Step 1, where the ciphertexts are computed using plaintext polynomial $P^{(\iota)}$ interpolated over the strings $\{h^1_{(i,\iota,l)}\}_{i\in[|X|],\iota\in[N],l\in[n]}$ rather than as $\{H_1(s_\iota, x_{(i-1)n+l})\}_{i\in[I],\iota\in[N],l\in[n]}$. This is possible because the semantic security of the homomorphic cryptosystem implies that the output of $\mathsf{Enc}(\cdot,\cdot)$ is indistinguishable to $\mathcal{A}$ on polynomials of degree n.

□

B.2 Malicious-case Secure Protocols for Other Functionalities

We describe a similar set of modifications to the protocol computing set intersection with labelling described in Appendix A and to the protocols computing the other functionalities. We omit the proofs of Theorems 10–12 as they run along similar lines to that of Theorem 9.

Theorem 10. *Assuming that H_1-H_3 are random oracles and that $\mathsf{Enc}(\cdot,\cdot)$ is a message encryption function of a semantically secure cryptosystem* SET-INTERSECTION-II* *achieves security against statically chosen malicious adversaries.*

Theorem 11. *Assuming that H_1-H_3 are random oracles and that $\mathsf{Enc}(\cdot,\cdot)$ is a message encryption function of a semantically secure cryptosystem* FUZZY-MATCHING* *achieves security against statically chosen malicious adversaries.*

Theorem 12. *Assuming that H_1 and H_2 are random oracles and that $\mathsf{Enc}(\cdot,\cdot)$ is a message encryption function of a semantically secure cryptosystem* CLOSEST-MATCHING* *achieves security against statically chosen malicious adversaries.*

procedure $\textsc{Set-Intersection-II}^*(X, Y, \ell(\cdot))$

  $\mathbf{P_1}, \mathbf{P_2}$: $k' = \max_{y\in Y} |\ell(y)|, s \leftarrow \max_\iota\{\frac{|X^{*(\iota)}|}{n}, |Y^{*(\iota)}|\}, t \leftarrow p_{\geq 2^{2k+\log_2(k'+\lambda)+1}} : t \equiv 1 \bmod 2s$

  $\mathbf{P_1}$:

    for $\iota \leftarrow 0$ to $N-1$ **do**

      Let $X^{*(\iota)} = \cup_{x\in X, i\in[k'+\lambda], b\in\{0,1\}} x\|i\|b$

      Enumerate $X^{*(\iota)}$ as $\{x_1, \ldots, x_{|X^{*(\iota)}|}\}$

      for $j \leftarrow 1$ to n **do**

        $I^{(\iota)} \leftarrow \frac{|X^{*(\iota)}|}{n}$

        Construct $\alpha^{(\iota,j)} = \sum_{i=0}^{I^{(\iota)}-1} a_i^{(\iota,j)} x^i$ such that $\alpha^{(\iota,j)}(H_1(s_\iota, x_{(j-1)I^{(\iota)}+l})) = 0 : l \in [I^{(\iota)}]$.

      end for

      $A^{(\iota)} \leftarrow (\mathsf{Enc}(\ldots, \mathsf{CRT}(a_i^{(\iota 1)}, \ldots, a_i^{(\iota,n)}), \mathsf{pk}), \ldots), \mathsf{pk}), \underbrace{\mathsf{Enc}^*(0, \mathsf{pk}), \ldots, \mathsf{Enc}^*(0, \mathsf{pk})}_{2s-I^{(\iota)}})$

    end for

    Send $(A^{(\iota)})_{\iota\in[N]}$.

    $J \leftarrow H_2((A^{(\iota)})_{\iota\in[N]})$.

    Send $(s_\iota)_{\iota\in[N]\setminus J}$.

    Send openings of $(A^{(\iota)})_{\iota\in J}$.

  $\mathbf{P_2}$:

    Verify openings and correctness of J, output $\perp$ if verification fails.

    Choose $w \in \mathbb{Z}_t^* : \mathrm{ord}_t(w) = 2s$

    for $\iota \in [N]\setminus J$ **do**

      Pick $t^{(\iota)} \in_R \{0,1\}^\lambda$.

      Let $Y^{*(\iota)} = \cup_{y\in Y, i\in[k'+\lambda]} y\|i\|(\ell(y)\|t^{(\iota)})_i$

      Enumerate $Y^{*(\iota)}$ as $\{y_1, \ldots, y_{|Y^{*(\iota)}|}\}$

      Construct $\beta^{(\iota)} = \sum_{i=0}^{|Y^{*(\iota)}|-1} b_i^{(\iota)} x^i$ such that $\beta^{(\iota)}(H_1(s_\iota, y_i)) = 0$ for $i \in |Y^{*(\iota)}|$.

      for $j \leftarrow 1$ to n **do**

        Let $\mu^{(\iota,j)}(x) = \sum_{i=0}^{s} m_i^{(\iota,j)} x^i$ where $m_i^{(\iota,j)} = H_3(t^{(\iota)}\|i\|j)$ for $i \in [n]$

        Let $\nu^{(\iota,j)}(x) = \sum_{i=0}^{s} n_i^{(\iota,j)} x^i$ where $n_i^{(\iota,j)} \in_R \mathbb{Z}_t$ for $i \in [s]$ and $j \in [n]$

        Compute $\upsilon^{(\iota,j)}(x) = \beta^{(\iota)}(x)\nu^{(\iota,j)}(x) = \sum_{i=0}^{2s-1} u_i^{(\iota,j)} x^i$, for $j \in [n]$.

      end for

      $A'^{(\iota)} \leftarrow \textsc{Hom-Transform}(A^{(\iota)}, 2s, \mathsf{false})$

      for $i \leftarrow 1$ to $2s$ **do**

        $T'^{(\iota)}_i \leftarrow A'^{(\iota)}_i \ltimes \mathsf{CRT}(\mu^{(\iota,1)}(w^{i-1}), \ldots, \mu^{(\iota,n)}(w^{i-1}))$

      end for

      $T^{(\iota)} \leftarrow \textsc{Hom-Transform}(T'^{(\iota)}, 2s, \mathsf{true})$

      $U^{(\iota)} \leftarrow (\mathsf{Enc}(\mathsf{CRT}(u_0^{(\iota,1)}, \ldots, u_0^{(\iota,n)}), \mathsf{pk}), \ldots, enc(\mathsf{CRT}(u_{2s-1}^{(\iota,1)}, \ldots, u_{2s-1}^{(\iota,n)}), \mathsf{pk}))$

      for $i \leftarrow 1$ to $2s$ **do**

        $\mathsf{ct}_i^{(\iota)} \leftarrow T_i^{(\iota)} + U_i^{(\iota)}$

      end for

      Send $(\mathsf{ct}_1^{(\iota)}, \ldots, \mathsf{ct}_{2s}^{(\iota)})_{\iota\in[N]\setminus J}$.

    end for

  $\mathbf{P_1}$:

    Decrypt and verify correctness of $(\mathsf{ct}_i^{(\iota)})_{i\in[2s], \iota\in[N]\setminus J}$.

end procedure

procedure FUZZY-MATCHING*$(X, Y, \ell(\cdot), D)$
  $\mathbf{P_2}$: Enumerate Y as $\{y_1, \ldots, y_{|Y|}\}$
  for $i \leftarrow 1$ to $|Y|$ **do**
    $\mathbf{P_2}$: FUZZY-MEMBERSHIP*$(X, y_i, \ell(y_i), D)$
  end for
end procedure

procedure FUZZY-MEMBERSHIP*$(X, y, \ell_y, D \in [k+1])$

$\mathbf{P_1}, \mathbf{P_2}: n' \leftarrow \frac{n}{4k+\lambda}, I \leftarrow \frac{|X|}{n'}, t \sim 2n \log 2n$

$\mathbf{P_1}$:

Enumerate X as $\{x_1, \ldots, x_{|X|}\}$

Choose $s_0, \ldots, s_{N-1} \in_R \{0,1\}^\lambda$

for $\iota \leftarrow 0$ to $N-1$ **do**

for $i \leftarrow 1$ to I **do**

for $j \leftarrow 1$ to $4k$ **do**

$\mathsf{ct}_i^{(\iota,j)} \leftarrow \mathsf{Enc}(\mathsf{CRT}(\underbrace{H_1(s_\iota, x_{(i-1)n'+1})^{(j)} \| .. \| H_1(s_\iota, x_{in'})^{(j)}}_{n'}), \mathsf{pk})$

end for

Send $(\mathsf{ct}_i^{(\iota,1)}, \ldots, \mathsf{ct}_i^{(\iota,4k)})$.

end for

end for

$J \leftarrow H_2((\mathsf{ct}_i^{(\iota,j)})_{i\in[I], j\in[4k], \iota\in[N]})$.

Send $(s_\iota)_{\iota\in[N]\setminus J}$.

Send openings of $(\mathsf{ct}_i^{(\iota,j)})_{i\in[I], j\in[4k], \iota\in J}$.

$\mathbf{P_2}$:

Verify openings and correctness of J, output $\perp$ if verification fails.

$\mathsf{ct}_{\mathrm{tgt}} \leftarrow \mathsf{Enc}^*(\mathsf{CRT}(\underbrace{0, \ldots, 2(D-1), 0, \ldots, 2(D-1), 0, 2, \ldots}_{n}), \mathsf{pk})$

for $\iota \in [N]\setminus J$ **do**

for $j \leftarrow 1$ to $4k$ **do**

$\mathsf{ct}_y^{(\iota,j)} \leftarrow \mathsf{Enc}^*(\mathsf{CRT}(\underbrace{H_1(s_\iota, y)^{(j)} \| .. \| H_1(s_\iota, y)^{(j)}}_{n'}), \mathsf{pk})$

end for

Pick $t^{(\iota)} \in_R \{0,1\}^\lambda$.

Write $\ell_y \| t^{(\iota)} = \ell_1 \| \ldots \| \ell_v$ where $v = \lfloor \frac{4k+\lambda}{D} \rfloor$

$\mathsf{ct}_{\ell_y}^{(\iota)} \leftarrow \mathsf{Enc}^*(\mathsf{CRT}(\underbrace{\overbrace{\ell_1, \ldots, \ell_1}^{D}, \ldots, \overbrace{\ell_v, \ldots, \ell_v}^{D}, \overbrace{0, \ldots, 0}^{4k+\lambda-vD} \overbrace{\ell_1, \ldots, \ell_1}^{D}, \ldots}_{n}), \mathsf{pk})$

for $i \leftarrow 1$ to I **do**

Write $H_3(t^{(\iota)}, i) = u_{i1}^{(\iota)} \| \ldots \| u_{in}^{(\iota)}$.

$p_i^{(\iota)} \leftarrow \mathsf{CRT}(u_{i1}^{(\iota)}, \ldots, u_{in}^{(\iota)})$

$\mathsf{ct}_i^{(\iota)} \leftarrow (\sum_{j=1}^{4k} (\mathsf{ct}_i^{(\iota,j)} - \mathsf{ct}_y^{(\iota,j)})^2 - \mathsf{ct}_{\mathrm{tgt}}) \ltimes p_i^{(\iota)} + \mathsf{ct}_{\ell_y}^{(\iota)}$

end for

end for

Send $(\mathsf{ct}_1^{(\iota)}, \ldots, \mathsf{ct}_I^{(\iota)})_{\iota\in[N]\setminus J}$.

$\mathbf{P_1}$:

Decrypt and verify correctness of $(\mathsf{ct}_i^{(\iota)})_{i\in[I], \iota\in[N]\setminus J}$.

end procedure

procedure CLOSEST-MATCHING*(x, Y)

  $\mathbf{P_1}, \mathbf{P_2}$: $M \leftarrow \sqrt{4k}, t \leftarrow 2$

  $\mathbf{P_1}$:

    Choose $s_0, \ldots, s_{N-1} \in_R \{0,1\}^\lambda$

    for $\iota \leftarrow 0$ to $N-1$ **do**

      Write $H_1(s_\iota, x) = x^{(\iota,1)} \| \ldots \| x^{(\iota,M)}$ with $|x^{(\iota,j)}| = \frac{4k}{M}$

      for $j \leftarrow 1$ to M **do**

        for $\text{msk} \in \{0,1\}^{\frac{4k}{M}}$ **do**

          $I^{(\iota)} \leftarrow \mathbb{1}_{|x^{(\iota,j)} \oplus \text{msk}|}$

          $W^{(\iota)}_{j,\text{msk}} \leftarrow (\mathsf{Enc}(I^{(\iota)}[0], \mathsf{pk}), \ldots, \mathsf{Enc}(I^{(\iota)}[\frac{4k}{M}], \mathsf{pk}))$

        end for

      end for

    end for

    Send $(W^{(\iota)}_{j,\text{msk}})_{j \in 1+[M], \text{msk} \in \{0,1\}^{\frac{4k}{M}}, \iota \in [N]}$.

    $J \leftarrow H_2((W^{(\iota)}_{j,\text{msk}})_{j \in 1+[M], \text{msk} \in \{0,1\}^{\frac{4k}{M}}, \iota \in [N]})$.

    Send $(s_\iota)_{\iota \in [N] \setminus J}$.

    Send openings of $(W^{(\iota)}_{j,\text{msk}})_{j \in 1+[M], \text{msk} \in \{0,1\}^{\frac{4k}{M}}, \iota \in J}$.

  $\mathbf{P_2}$:

    Enumerate Y as $\{y_1, \ldots, y_{|Y|}\}$

    Verify openings and correctness of J, output $\perp$ if verification fails.

    for $\iota \in [N] \setminus J$ **do**

      for $i \leftarrow 1$ to $|Y|$ **do**

        Write $H_1(s_\iota, y_i) = y_i^{(\iota,1)} \| \ldots \| y_i^{(\iota,M)}$

        for $j \leftarrow 1$ to M **do**

          $P^{(\iota)}_{ij} \leftarrow \text{EXPAND}(W^{(\iota)}_{j, y_i^{(\iota,j)}}, 2k+1)$

        end for

        $P^{(\iota)}_i \leftarrow \prod_{j=1}^{M} P^{(\iota)}_{ij}$

        $W^{(\iota)}_i \leftarrow P^{(\iota)}_i[1]$

        $\mathsf{ct}^{(\iota)}_i \leftarrow \mathsf{Enc}(\mathsf{CRT}(y_i[1], \ldots, y_i[4k]), \mathsf{pk})$

      end for

      $N \leftarrow \frac{|Y|}{2}$

      for $h \leftarrow 0$ to $\log_2 |Y| - 1$ **do**

        for $i \leftarrow 1$ to N **do**

          $\mathsf{ct}^{(\iota)}_{(<)} \leftarrow \sum_{(a,b):b>a} W^{(\iota)}_i[a] \cdot W^{(\iota)}_{i+2^h}[b]$

          $\mathsf{ct}^{(\iota)}_i \leftarrow \mathsf{ct}^{(\iota)}_{i+2^h} + \mathsf{ct}^{(\iota)}_{(<)} \cdot (\mathsf{ct}^{(\iota)}_i - \mathsf{ct}^{(\iota)}_{i+2^h})$

          $W^{(\iota)}_i \leftarrow W^{(\iota)}_{i+2^h} + \mathsf{ct}^{(\iota)}_{(<)} \cdot (W^{(\iota)}_i - W^{(\iota)}_{i+2^h})$

          $N \leftarrow \frac{N}{2}$

        end for

      end for

    end for

    Send $(\mathsf{ct}^{(\iota)}_1)_{\iota \in [N] \setminus J}$.

  $\mathbf{P_1}$:

    Decrypt and verify correctness of $(\mathsf{ct}^{(\iota)}_1)_{\iota \in [N] \setminus J}$.

end procedure