Communication Optimal Unbalanced Private Set Union

Jean-Guillaume Dumas, Alexis Galan, Bruno Grenet, Aude Maignan, Daniel S. Roche

Communication Optimal Unbalanced Private Set Union

Jean-Guillaume Dumas*[1], Alexis Galan†[1], Bruno Grenet‡[1], A. Maignan §[1], and Daniel S. Roche ¶[1]

[1]Univ. Grenoble Alpes,CNRS,Grenoble INP, LJK, UMR CNRS 5224, 38000 Grenoble, France
[2]United States Naval Academy, Annapolis, Maryland, United States

Abstract

We consider the private set union (PSU) problem, where two parties each hold a private set of elements, and they want one of the parties (the receiver) to learn the union of the two sets and nothing else. Our protocols are targeted for the unbalanced case where the receiver's set size is larger than the sender's set size, with the goal of minimizing the costs for the sender both in terms of communication volume and local computation time. This setting is motivated by applications where the receiver has significantly more data (input set size) and computational resources than the sender which might be realized on a small, low-power device. Asymptotically, we achieve communication cost linear in the sender's (smaller) set size, and computation costs for sender and receiver which are nearly-linear in their respective set sizes. To our knowledge, ours is the first algorithm to achieve nearly-linear communication and computation for PSU in this unbalanced setting. Our protocols utilize fully homomorphic encryption (FHE) and, optionally, linearly homomorphic encryption (LHE) to perform the necessary computations while preserving privacy. The underlying computations are based on univariate polynomial arithmetic realized within homomorphic encryption, namely fast multiplication, modular reduction, and multi-point evaluation. These asymptotically fast HE polynomial arithmetic algorithms may be of independent interest.

1 Introduction

A Private Set Union (PSU) protocol is a cryptographic protocol involving two parties, in which a receiver, denoted $\mathcal{R}$, owns a set $\mathbf{X}$, and a sender, denoted $\mathcal{S}$, owns a set $\mathbf{Y}$. The functionality desired from such a protocols is denoted $\mathcal{F}_{PSU}$ and is presented in Func. 1: the receiver $\mathcal{R}$ receives the union $\mathbf{X} \cup \mathbf{Y}$. The protocol is parameterized on (upper bounds on) the set sizes $|X|$ and $|Y|$, which are therefore implicitly revealed to both parties as well. However, the sender $\mathcal{S}$ learns nothing about the contents of $\mathbf{X}$.

We are interested in the case of unbalanced inputs, where the sender and receiver set sizes may be (vastly) different, and on minimizing the communication volume between the two parties. Note that it is impossible

*Jean-Guillaume.Dumas@univ-grenoble-alpes.fr
†Alexis.Galan@univ-grenoble-alpes.fr
‡Bruno.Grenet@univ-grenoble-alpes.fr
§Aude.Maignan@univ-grenoble-alpes.fr
¶Roche@usna.edu

Functionality 1: $\mathcal{F}_{PSU}$, Private Set Union

$\mathcal{R}$	$\mathbf{X} \rightarrow$ $\mathbf{X} \cup \mathbf{Y} \leftarrow$	PSU Protocol	$\leftarrow \mathbf{Y}$	$\mathcal{S}$

to reduce the communication size below that of the sender's set, since in the worst case this entire set must be revealed to the receiver, and the sender must not know how many elements were actually revealed.

The more difficult situation for unbalanced sets is therefore when the sender's set size is smaller than the receiver's, and we may hope to have sub-linear worst-case communication costs; this is the focus of our work. More precisely, if we define m and n to be respectively the sizes of the sender's and receiver's sets, we will assume $n \geq m$ in our setting, and our goal is to develop a PSU protocol with $O(m)$ communication (which we can achieve) and $O(m)$ and $O(n)$ computation resp. for the sender and receiver (which we can *nearly* achieve).

The recent surge in research for efficient PSU protocols is motivated by numerous practical applications. One example which is motivating for our unbalanced setting with larger receiver is secure aggregation as in [Ramanathan et al.(2020)]: A single server maintains a growing list, and individual contributors periodically interact with the server to add their elements into the list. If the collected list may be sensitive, such as individuals who may have been exposed to some communicable disease, then the server may not want to reveal the list to every contributor; similarly, the contributors may not wish to reveal their entries which are already on the list to avoid potential inferences on relationships between the contributors themselves.

Table 1: Protocol Comparison Table: receiver $\mathcal{R}$ set size n, sender $\mathcal{S}$ set size m, with $n \geq m$

Protocol	**Frikken** [Frikken(2007)]	**Dav. & Cid.** [Davidson and Cid(2017)]	**Zhang et al.** [Zhang et al.(2023)]	**Tu et al.** [Tu et al.(2023)]	**Our UPSU**
Arith. cost for $\mathcal{R}$	$O\left(n^{1+\epsilon}\right)$	$O(n)$	$O(n)$	$O(n)$	$O\left(n^{1+\epsilon}\right)$
Arith. cost for $\mathcal{S}$	$O(nm)$	$O(m)$	$O(m \log n)$	$O\left(m^2\right)$	$O\left(m^{1+\epsilon}\right)$
Comm. volume	$O(n)$	$O(n)$	$O(n)$	$O(m \log n)$	$O(m)$
Deterministic	✓	✗	✓	✗	✓

Previous work. Privacy-preserving set operations have traditionally started with private set intersection (PSI), which has seen a number of recent efficient protocols and important applications such as private contact discovery [Kiss et al.(2017), Groce et al.(2019), Resende and de Freitas Aranha(2021), Gordon et al.(2022), Badrinarayanan et al.(2022), Morales et al.(2023)].

Private set union protocols have also garnered significant recent interest [Brickell and Shmatikov(2005), Kissner and Song(2005), Frikken(2007), Davidson and Cid(2017), Kolesnikov et al.(2019), Garimella et al.(2021), Jia et al.(2022), Zhang et al.(2023), Tu et al.(2023)]. We mention a few results most closely related to the current work.

Frikken's PSU algorithm [Frikken(2007)] represents a set as a polynomial, whose roots are the set elements. Then, the elements of the sender that are not roots of the receiver's polynomial, are exactly the elements that must be exchanged. This protocol requires a polynomial evaluation on the receiver's polynomial in all the sender's elements, and the receiver should learn nothing from a root but should retrieve the evaluated elements from a non zero. The Paillier linearly homomorphic (LHE) scheme is used to keep the receiver's set private.

Instead of using polynomials and its zeroes, Davidson and Cid [Davidson and Cid(2017)] proposed a version using Bloom filters and its zeroes, also LHE encrypted, in order to hide the receiver's set. This improves asymptotically on on Frikken's protocol, but the usage of Bloom filters makes it non deterministic.

In [Zhang et al.(2023)], Zhang et al. imagined a version divided in two sub-protocols. The first one, called multi-query reverse private membership test, gives to the receiver a bit-vector where the zeroes represent the elements of the sender that are not in the receiver's set. The second sub-protocol is an oblivious transfer (OT) where the bit-vector represent the choice-bits. All those protocols where not designed for an unbalanced situation and their communication volume is always proportional to the size of the receiver's set.

To our knowledge, the best private set union protocol specially designed for the unbalanced situation comes from [Tu et al.(2023)] where Tu et al. proposed a protocol using many different techniques (set hashing, windowing, baby-set-giant-step precomputation matrix, fully homomorphic encryption scheme (FHE), oblivious transfer...) in order to reduce the communication volume to a logarithmic dependency in the size of the receiver's set. The usage of Cuckoo hashing makes this protocol non-deterministic as well.

Our contributions. We present two new (related) protocols for unbalanced PSU (thus UPSU), first a generic one, and then an instantiation of it:

- Our generic UPSU protocol is in two parts and relies on efficient polynomial arithmetic algorithms, such as polynomial remainder and polynomial multipoint evaluation, that applied over FHE and LHE schemes. This protocol requires that the plaintext spaces of the FHE and LHE are compatible.
- We show that an instantiation of our protocol is possible with the BGV cryptosytem used separately for both parts, as both FHE and LHE.

The security of our protocol is proved in the semi-honest setting, and our complexity analysis is in the arithmetic setting (equivalently, assuming that all input elements are constant-sized).

Table 1 summarizes the cost analysis of the previously mentioned protocols in an unbalanced situation: the receiver $\mathcal{R}$ owns a set of size n, the sender $\mathcal{S}$ owns a set of size m with $n \geq m$. In the table, the "Comm. volume" row represents a bound on the quantity of elements exchanged, and the "Arith. cost" rows represent a bound on the number of basic arithmetic operations done by each party. A value colored in green is a value satisfying our goals, which are an arithmetic cost for the sender and a communication volume independent of the size of the receiver's (larger) set and a deterministic algorithm. Orange and red values are used to denote larger dependencies, that is, logarithmic, or more, respectively, in the size of the larger set.

Outline. In Section 2, we present the blocks needed to build our protocols, including our algorithms for polynomial arithmetic over LHE and FHE. Section 3 defines the security expected from our protocols and lists the security assumptions we are making. Our unbalanced PSU, using generic LHE and FHE schemes, in presented, proven correct and secure under honest-but-curious adversary model and analyzed in term of its asymptotics in Section 4. We show that we can instantiate this protocol, with the BGV cryptosystem both as LHE and FHE, and we compare our simulated communication volume to [Tu et al.(2023)] in Section 5.

2 Building Blocks

In this section, we present our main building blocks based on polynomial arithmetic. We show that we can perform efficient polynomial arithmetic homomorphically. We distinguish between tasks that require a fully homomorphic encryption scheme and those who can be implemented within a linearly homomorphic encryption scheme. Since it has an important impact on the practical efficiency, we also study the multiplicative depth of these algorithms.

2.1 Cryptographic tools

We first introduce the main crytographic tools our protocol is based on, namely Linearly and Fully Homomorphic Encryption Schemes.

2.1.1 Homomorphic Encryption Scheme (LHE)

For our purposes[1], a linearly homomorphic encryption (LHE) scheme consists of five algorithms

$$(\mathbf{L.Setup}, \mathbf{L.E}, \mathbf{L.D}, +_L, \ltimes_L) :$$

- $(pk, sk) \leftarrow \mathbf{L.Setup}(\kappa)$: Given a security parameter κ, outputs a pair of secret and public keys (pk, sk). pk implicitly defines a ring $\mathbb{M}$, which is the plaintext space, and a ciphertext space $\mathbb{E}$;
- $c \leftarrow \mathbf{L.E}_{pk}(m)$: Given as inputs a plaintext $m \in \mathbb{M}$ and a public key pk, outputs a ciphertext $c \in \mathbb{E}$;
- $m \leftarrow \mathbf{L.D}_{sk}(c)$: Given as inputs a ciphertext $c \in \mathbb{E}$ and a public key sk, outputs a plaintext $m \in \mathbb{M}$;
- $c_3 \leftarrow c_1 +_L c_2$: Given as inputs two ciphertexts $c_1, c_2 \in \mathbb{E}$, outputs a ciphertext $c_3 \in \mathbb{E}$;
- $c_3 \leftarrow m_1 \ltimes_L c_2$: Given as inputs a plaintext $m_1 \in \mathbb{M}$ and a ciphertext $c_2 \in \mathbb{E}$, outputs a ciphertext $c_3 \in \mathbb{E}$.

Definition 1. $(\mathbf{L.Setup}, \mathbf{L.E}, \mathbf{L.D}, +_L, \ltimes_L)$ is a semantically secure LHE if it satisfies the following properties:

i) **Correctness.** For any security parameter κ, if $(pk, sk) \leftarrow \mathbf{L.Setup}(\kappa)$, for all $m, m_1, m_2 \in \mathbb{M}$,

$$\mathbf{L.D}_{sk}(\mathbf{L.E}_{pk}(m)) = m, \tag{1}$$
$$\mathbf{L.D}_{sk}(\mathbf{L.E}_{pk}(m_1) +_L \mathbf{L.E}_{pk}(m_2)) = m_1 + m_2, \tag{2}$$
$$\mathbf{L.D}_{sk}(m_1 \ltimes_L \mathbf{L.E}_{pk}(m_2)) = m_1 m_2. \tag{3}$$

ii) **Security.** The scheme is semantically secure if it is not possible to derive from a ciphertext more than negligible information on the plaintext.

2.1.2 Fully Homomorphic Encryption Scheme (FHE)

A fully homomorphic encryption (FHE) scheme consists of six algorithms

$$(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F) \tag{4}$$

where $(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F)$ is a LHE and the algorithm $\times_F$ is as follows:

- $c_3 \leftarrow c_1 \times_F c_2$: Given as inputs two ciphertexts c_1, $c_2 \in \mathbb{E}$, outputs a ciphertext $c_3 \in \mathbb{E}$.

Definition 2. $(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F)$ is a semantically secure FHE if it satisfies the following properties:

[1] More generally LHE may be defined over only a group and not a ring, but we need in particular plaintext-ciphertext multiplications over a ring for our application here.

i) **Correctness.** For any security parameter κ, (**F.Setup**, **F.E**, **F.D**, $+_F, \ltimes_F$) satisfies the LHE correctness and if $(pk, sk) \leftarrow$ **F.Setup**(κ), for all $m_1, m_2 \in \mathbb{M}$,

$$\mathbf{F.D}_{sk}(\mathbf{F.E}_{pk}(m_1) \times_F \mathbf{F.E}_{pk}(m_2)) = m_1 m_2. \tag{5}$$

ii) **Security.** The scheme is semantically secure if it is not possible to derive from a ciphertext more than negligible information on the plaintext.

Remark 3. We extend the encryption and decryption algorithms for a LHE or a FHE to allow vectors as inputs: if $v \in \mathbb{M}^n$, $\mathbf{L.E}_{pk}(v)$ (resp. $\mathbf{F.E}_{pk}(v)$) outputs $c \in \mathbb{E}^n$ such that c_i is the encryption for v_i for $1 \leq i \leq n$. Similarly, $\mathbf{L.D}_{sk}(c)$ (resp. $\mathbf{F.D}_{sk}(c)$) outputs v. In the same way, we extend these algorithms to polynomial inputs and outputs in $\mathbb{M}[X]$ or $\mathbb{E}[X]$ by stating that the encryption of a polynomial is the encryption of its vector of coefficients.

This allows for instance to extend the algorithm $+_L$ (resp. $+_F$) to vectors or polynomials. Also, we can extend $\ltimes_L$ (resp. $\ltimes_F$) to a matrix-vector product where the matrix is in clear and the vector encrypted:

$$\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & & \vdots \\ v_{m1} & \cdots & v_{mn} \end{pmatrix} \ltimes_L \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} v_{11} \ltimes_L c_1 +_L \cdots +_L v_{1n} \ltimes_L c_n \\ \vdots \\ v_{m1} \ltimes_L c_1 +_L \cdots +_L v_{mn} \ltimes_L c_n \end{pmatrix} \tag{6}$$

The resulting algorithm has multiplicative depth 1.

Notations Since we use both a LHE scheme and a FHE scheme, for clarity, we will denote with $\widehat{x}$ a variable which is encrypted under a LHE scheme, and with $\widetilde{x}$ a variable which is encrypted under a FHE scheme.

2.2 Linearly homomorphic polynomial arithmetic

In this section, we focus on polynomial operations than can be performed linearly homomorphically, when one of the inputs is in clear.

As customary in polynomial arithmetic, the homomorphic algorithms we describe reduce to polynomial multiplications. For efficiency considerations we first prove that faster-than-quadratic polynomial multiplication algorithms can be performed in linearly homorphic settings. For $P \in \mathbb{M}[X]$ and $C \in \mathbb{E}[X]$, we let $P \ltimes_L C \in \mathbb{E}[X]$ be the encrypted polynomial such that $\mathbf{L.D}_{sk}(P \ltimes_L C) = P \times \mathbf{L.D}_{sk}(C)$.

In the following, we denote by $\mathcal{M}_L(d)$ the arithmetic cost of a linearly homomorphic product between a clear polynomial and an encrypted one, both of degrees at most d. As an example, the following lemma shows that for any LHE, we can build a Toom-k algorithm for the homomorphic product $P \ltimes_L C$. Similar result can be proved for FFT-based multiplication algorithms, provided the plaintext space contains suitable roots of unity.

Lemma 4. *If $Toom_k(d)$ denotes the arithmetic cost of a Toom-k algorithm on polynomials of degree d,*

$$\mathcal{M}_L(d) = O(Toom_k(d)). \tag{7}$$

Proof. A Toom-k algorithm basically requires the product of a Vandermonde matrix by a vector for polynomial interpolation and evaluation. To compute a linearly homomorphic product $P \ltimes \widehat{Q}$ where $\widehat{Q}$ is encrypted, we can build a Vandermonde matrix representing the powers of clear evaluation points, and we can consider

both P and $\widehat{Q}$ as vectors. The evaluations of P et $\widehat{Q}$ are given by matrix-vector products (in clear and homorphically using (6), resp.). We can perform a clear/encrypted pointwise multiplication on the evaluation vectors with $\ltimes_L$ as one vector is in clear. Finally, the interpolation is done by another matrix-vector product between the inverse Vandermonde and the encrypted vector obtained. □

Our algorithms use of the *middle product* of two polynomials, that we now define. For a polynomial $P = \sum_{i=0}^{d} p_i X^i \in \mathbb{M}[X]$, let $\overleftarrow{P} := \sum_{i=0}^{d} p_{d-i} X^i = P(\frac{1}{X})X^d$ be its *reverse polynomial*, and for $a \leq b \leq d$, let $[P]_a^b := \sum_{i=a}^{b} p_i X^{i-a}$. The *middle product* of two polynomials $P = \sum_{i=0}^{d} p_i X^i$ and $Q = \sum_{i=0}^{d'} q_i X^i$ is defined to be

$$[\overleftarrow{P} Q]_d^{d'} = \sum_{i=0}^{d'-d} \sum_{j=0}^{d} p_j q_{j+i} X^i. \tag{8}$$

Definition 5. Let $(\mathbf{L.Setup}, \mathbf{L.E}, \mathbf{L.D}, +_L, \ltimes_L)$ be a LHE scheme. The linearly homomorphic middle product is an algorithm **L.Mid** satisfying

- $C_2 \leftarrow \mathbf{L.Mid}(P, C_1)$: Given as inputs a clear polynomial $P \in \mathbb{M}[X]$ and an encrypted polynomial $C_1 \in \mathbb{E}[X]$, outputs an encrypted polynomial $C_2 \in \mathbb{E}[X]$.

This algorithm satisfies the following correctness property.

Correctness. For a security parameter κ, for every P_1, $P_2 \in \mathbb{M}[X]$, with m and M the minimum and the maximum degrees of those polynomials respectively, and for $(pk, sk) \leftarrow \mathbf{L.Setup}(\kappa)$,

$$\mathbf{L.D}_{sk}\left(\mathbf{L.Mid}\left(P_1, \mathbf{L.E}_{pk}\left(P_2\right)\right)\right) = \left[\overleftarrow{P_1} P_2\right]_m^M. \tag{9}$$

To obtain an algorithm for **L.Mid**, we rely on Tellegen's transposition principle [Bostan et al.(2003)]. For, we view the multiplication between $P \ltimes_L C$ as a linear map by fixing the clear polynomial P. The transposed of this linear map is exactly the linearly homomorphic middle product, again with the clear polynomial fixed. General results on transposition guarantee that an algorithm for computing $P \ltimes_L C$ using t arithmetic operations can be *transposed* to get an algorithm for **L.Mid** for polynomials of degree n and $n+m$, that uses $t+n$ arithmetic operations where $n = \deg(P)$ and $m = \deg(C)$. This implies the following lemma.

Lemma 6. *Let $P \in \mathbb{M}[X]$ of degree n and $C \in \mathbb{E}[X]$ of degree $n+m$. Then **L.Mid**(P, C) can be computed in $\mathcal{M}_L(\max(n, m)) + O(m)$ operations, with constant multiplicative depth.*

We now turn to one of our main building blocks: multipoint evaluation. Given a degree-d polynomial P and k evaluation points $m_1, \ldots, m_k$, it consists in evaluating P on each m_i.

Definition 7. Let $(\mathbf{L.Setup}, \mathbf{L.E}, \mathbf{L.D}, +_L, \ltimes_L)$ be a LHE scheme. The linearly homomorphic multipoint evaluation is an algorithm **L.MultEv** such that

- $\{c_1, \ldots, c_k\} \leftarrow \mathbf{L.MultEv}(C, \{m_1, \ldots, m_k\})$: Given as inputs an encrypted polynomial $C \in \mathbb{E}[X]$ and a set of k plaintexts $m_1, \ldots, m_k \in \mathbb{M}$, outputs a set of k ciphertexts $c_1, \ldots, c_k \in \mathbb{E}$.

The algorithm satisfies the following correctness property.

Correctness. For a security parameter κ, for every $P \in \mathbb{M}[X]$, every subset $\{m_1, \ldots, m_k\} \subset \mathbb{M}$ and $(pk, sk) \leftarrow \mathbf{L.Setup}(\kappa)$, if

$$\{c_1, \ldots, c_k\} \leftarrow \mathbf{L.MultEv}\left(\mathbf{L.E}_{pk}\left(P\right), \{m_1, \ldots, m_k\}\right) \tag{10}$$

then, for all $i \in \{1, \ldots, k\}$,

$$\mathbf{L.D}_{sk}(c_i) = P(m_i). \tag{11}$$

Algorithm **L.MultEv** can be implemented in $O(dk)$ operations where $d = \deg(P)$, evaluating P on each point with Horner scheme. The multiplicative depth of this algorithm is 1. We now turn to an asymptotically fast algorithm when $k = d + 1$.

Proposition 8. *Let $C \in \mathbb{E}[X]$ of degree d and $m_0, \ldots, m_d \in \mathbb{M}$ be evaluation points. **L.MultEv**$(C, \{m_0, \ldots, m_d\})$ can be computed in $\mathcal{M}_L(d) \log d + O(d \log d)$ operations, after $\frac{1}{2}\mathcal{M}(d) \log d + O(d \log d)$ operations of precomputation on $m_0, \ldots, m_d$. The multiplicative depth of the computation is $O(\log d)$.*

Proof. We adapt the algorithm presented in [Bostan et al.(2003)] to the LHE context. Let $C = \sum_{i=0}^{d-1} c_i X^i$, $P = \mathbf{L.D}_{sk}(C)$. and $m_0, \ldots, m_d \in \mathbb{M}$. We assume that d is as power of two to ease the description of the algorithm, but it is not mandatory in practice. The first step of the algorithm consists in computing the following polynomials in clear, for $k = 0, \ldots, \log d$ and $i = 1 \ldots, 2^k$:

$$P_{\left(\frac{i}{2^k}\right)} := \prod_{j \in \left\{\frac{i-1}{2^k}d+1, \ldots, \frac{i}{2^k}d\right\}} (X - m_j) \tag{12}$$

These polynomials can be computed using a product tree in
$\frac{1}{2}\mathcal{M}(d) \log d + O(d \log d)$ arithmetic operations. Note that these polynomials can be precomputed if the evaluation points are known in advance.

The algorithm requires then to compute the polynomials

$$B := \overleftarrow{P_{\left(\frac{1}{1}\right)}}^{-1} \mod X^d, \text{ and} \tag{13}$$

$$A := \left[\overleftarrow{B} \ltimes_L C\right]_{d-1}^{2d-1}. \tag{14}$$

Let $A_{\left(\frac{1}{1}\right)} := \overleftarrow{A}$. The last step of the algorithm consists in the computation for $k = 1, \ldots, \log d$ and $i = 1, \ldots, 2^k$ of the encrypted polynomials

$$A_{\left(\frac{i}{2^k}\right)} = \mathbf{L.Mid}\left(P_{\left(\frac{i-(-1)^{(i \mod 2)}}{2^k}\right)}, A_{\left(\frac{\lceil i/2 \rceil}{2^{k-1}}\right)}\right). \tag{15}$$

According to the correctness of the algorithm presented in [Bostan et al.(2003)], $A_{\left(\frac{i}{d}\right)}$ is an encryption of $P(m_i)$ for $1 \leq i \leq d$. The final computation of the polynomials $A_{\left(\frac{i}{2^k}\right)}$ requires $\mathcal{M}_L(d) \log d + O(d \log d)$ arithmetic operations, and this dominates the cost. □

2.3 Fully homomorphic polynomial arithmetic

Another building block of our protocol is the computation of a polynomial remainder. This is a harder task in the context of homomorphic encryption. In our case, we divide a clear polynomial by an encrypted one. This computation cannot be performed in a LHE scheme since the divisor and the quotient, both encrypted, need to be multiplied together. Moreover, the need to invert the leading coefficient of the divisor could be a problem. We focus here on the case where the divisor is monic.

The standard algorithm for this task is the quadratic long division algorithm. This algorithm has a linear multiplicative depth. Below we show how to adapt the fast euclidean division algorithm, based on Newton iteration, to the FHE settings. The algorithm is quasi-linear and has only a logarithmic multiplicative depth.

To describe it, we need to a polynomial multiplication algorithm in the FHE. We denote by $\mathcal{M}_F(d)$ the arithmetic cost of a homomorphic product $C_1 \times_F C_2$ between two encrypted polynomials of degrees at most d in FHE. The same argument as in the LHE case shows that we can adapt faster-than-quadratic algorithms for polynomial multiplication to the FHE settings, with constant multiplicative depth.

Definition 9. Let $(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F)$ be a FHE scheme. The homomorphic polynomial remainder is an algorithm **F.Rem** as follows:

- $C_3 \leftarrow \mathbf{F.Rem}(C_1, C_2)$: Given as inputs two encrypted polynomials C_1, $C_2 \in \mathbb{E}[X]$, outputs an encrypted polynomial $C_3 \in \mathbb{E}[X]$.

This algorithm satisfies the following correctness property.

Correctness. For a security parameter κ, for $(pk, sk) \leftarrow \mathbf{F.Setup}(\kappa)$, and for clear polynomials P_1, $P_2 \in \mathbb{M}[X]$ such that P_2 is monic,

$$\mathbf{F.D}_{sk}\left(\mathbf{F.Rem}\left(\mathbf{F.E}_{pk}\left(P_1\right), \mathbf{F.E}_{pk}\left(P_2\right)\right)\right) = P_1 \bmod P_2. \tag{16}$$

Proposition 10. *Let C_1, $C_2 \in \mathbb{E}[X]$ of respective degrees n and $m < n$, where C_2 is the encryption of a monic polynomial, **F.Rem**(C_1, C_2) can be computed in at most $\frac{9}{2}\mathcal{M}_F(n-m) + O(n-m)$ arithmetic operations, with a multiplicative depth $O(\log(n-m))$.*

Proof. We recall the Netwon-iteration-based algorithm for polynomial euclidean division. We present the fast version based on middle products. The remainder R in the division of A by B, of respective degrees n and $m < n$, is the unique polynomial satisfying $A = BQ + R$ with $\deg(R) < m$. This implies $\overleftarrow{A} = \overleftarrow{Q}\overleftarrow{B} + X^{n-m+1}\overleftarrow{R}$, whence

$$\overleftarrow{Q} = \overleftarrow{A}\,\overleftarrow{B}^{-1} \mod X^{n-m+1}. \tag{17}$$

The goal is to homomorphically compute the inverse of $\overleftarrow{B}$ modulo X^{n-m+1}, using Newton iteration. Let C_A and C_B be the encryptions of A and B, and $\widetilde{1}$ be an encryption of 1 with the same public key. The algorithm requires first to compute the $t+1 := \lceil \log(n-m+1) \rceil$ first polynomials of the sequence (U):

$$(U) = \begin{cases} U_0 = \widetilde{1} \\ U_{k+1} = U_k \times_F \left(\widetilde{1} -_F \left[\overleftarrow{C}_B \times_F U_k\right]_{2^k}^{2^{k+1}-1} X^{2^k}\right) \mod X^{2^{k+1}} \end{cases}$$

Now, instead of computing the last step of the sequence that would give us homomorphically the inverse polynomial of $\overleftarrow{B} \mod X^{n-m+1}$, we directly compute the quotient, homomorphically. Let U_t be the $(t+1)^{\text{st}}$ polynomial of this sequence. We compute

$$S = \overleftarrow{C_A} \times_F U_t \mod X^{n-m+1}, \text{ and} \tag{18}$$

$$T = \left[\overleftarrow{C_B} U_t\right]_{2^k}^{2^{k+1}-1} \times_F [S]_0^{n-m-2^k} \mod X^{n-m+1-2^k}. \tag{19}$$

Then $\overleftarrow{C_Q} := S +_F TX^{2^k}$ is an encryption of $\overleftarrow{Q}$, the reverse quotient. Finally, we compute

$$C_R = C_A -_F C_Q \times_F C_B \mod X^m \tag{20}$$

to get an encryption of the remainder R. Using the fact that $\mathcal{M}_F(2d) \leq 2\mathcal{M}_F(d)$, we can bound the number of arithmetic operations done with that algorithm with $\frac{9}{2}\mathcal{M}_F(n-m) + O(n-m)$. The multiplicative depth is $O(t)$. □

3 Security Model and Assumptions

3.1 Security Model

We are following the definition of security for a two-party protocol presented in [Lindell(2017)]. Let Π be a two-party protocol computing a polynomial-time functionality $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2)$. For x and y, inputs of each party, the *ideal output-pair* is $f(x,y) = (f_1(x,y), f_2(x,y))$ where party i outputs $f_i(x,y)$. The *view* of the i-th party with inputs (x,y) is the tuple

$$\mathbf{view}_i^{\Pi}(x,y) := (w, C_i, M_i) \tag{21}$$

where w is the i-th party's input, C_i regroups all the information generated or computed by the i-th party during the protocol and M_i is the content of the messages received by this party during the protocol. The *output* of the i-th party with inputs (x,y) is denoted $\mathbf{output}_i^{\Pi}(x,y)$ and can be computed from $\mathbf{view}_i^{\Pi}(x,y)$. The *joint output* is denoted

$$\mathbf{output}^{\Pi}(x,y) = (\mathbf{output}_1^{\Pi}(x,y), \mathbf{output}_2^{\Pi}(x,y)) \tag{22}$$

Definition 11. Let $f = (f_1, f_2)$ be a functionality. We say that π *securely computes* f *in the presence of honest-but-curious adversaries* if there exists a probabilistic polynomial-time algorithms S_1 and S_2 such that for any finite set of inputs $I, J \subset \{0,1\}^*$:

$$\{S_1(x, f_1(x,y)), f(x,y)\}_{I,J} \stackrel{c}{\equiv} \{\mathbf{view}_1^{\Pi}(x,y), \mathbf{output}^{\Pi}(x,y)\}_{I,J}$$
$$\{S_2(y, f_2(x,y)), f(x,y)\}_{I,J} \stackrel{c}{\equiv} \{\mathbf{view}_2^{\Pi}(x,y), \mathbf{output}^{\Pi}(x,y)\}_{I,J}$$

where $\stackrel{c}{\equiv}$ denotes the computational indistinguishability.

3.2 Unbalanced Private Set Union Scheme

An unbalanced private set union scheme (UPSU) consists of five algorithms: **Setup**, **Y.Enc**, **X.ExtractEnc**, **Comput**, **Union** between a sender $\mathcal{S}$ that owns a set $\mathbf{Y} \subset \mathbb{M}$ and a receiver $\mathcal{R}$ that owns a set $\mathbf{X} \subset \mathbb{M}$.
• $\{keys_{\mathcal{R}}, keys_{\mathcal{S}}\} \leftarrow$**Setup** (κ): On input of a security parameter κ, outputs receiver's keys $keys_{\mathcal{R}}$ and sender's keys $keys_{\mathcal{S}}$.
• $E_{\mathbf{Y}} \leftarrow$**Y.Enc** $(\mathbf{Y}, keys_{\mathcal{S}})$: Given sender's set $\mathbf{Y}$ and keys $keys_{\mathcal{S}}$, outputs $E_{\mathbf{Y}}$, an encoding of the set $\mathbf{Y}$.
• $E_{\mathbf{X}} \leftarrow$**X.ExtractEnc** $(\mathbf{X}, keys_{\mathcal{R}}, E_{\mathbf{Y}})$: As input, takes receiver's set $\mathbf{X}$, keys $keys_{\mathcal{R}}$ and $E_{\mathbf{Y}}$, an encoding of the set $\mathbf{Y}$. Outputs $E_{\mathbf{X}}$, a (partial) encoding of the set $\mathbf{X}$.
• $\mathcal{D} \leftarrow$**Comput** $(E_{\mathbf{Y}}, E_{\mathbf{X}}, keys_{\mathcal{S}})$: On input of $E_{\mathbf{Y}}$ and $E_{\mathbf{X}}$, the encoding of each set, and sender's keys $keys_{\mathcal{S}}$, outputs a data set $\mathcal{D}$.
• $\mathbf{Z} \leftarrow$**Union** $(\mathbf{X}, \mathbf{Y}, \mathcal{D}, keys_{\mathcal{R}})$: On input of the receiver's set $\mathbf{X}$, the sender's set $\mathbf{Y}$, a data set $\mathcal{D}$ and receiver's keys $keys_{\mathcal{R}}$, outputs a set $\mathbf{Z}$.

Definition 12. (**Setup**, **Y.Enc**, **X.ExtractEnc**, **Comput**, **Union**) is a secure unbalanced private set union scheme under honest-but-curious adversary model if it satisfies the following three properties:

i) **Correctness.** For a security parameter κ and any sets $\mathbf{X}, \mathbf{Y} \subset \mathbb{M}$, for

$$\begin{aligned} \{keys_{\mathcal{R}}, keys_{\mathcal{S}}\} &\leftarrow \mathbf{Setup}(\kappa) \\ E_{\mathbf{Y}} &\leftarrow \mathbf{Y.Enc}(\mathbf{Y}, keys_{\mathcal{S}}) \\ E_{\mathbf{X}} &\leftarrow \mathbf{X.ExtractEnc}(\mathbf{X}, keys_{\mathcal{R}}, E_{\mathbf{Y}}) \end{aligned}$$

then the scheme is correct if:

$$\mathbf{Union}(\mathbf{X}, \mathbf{Y}, \mathbf{Comput}(E_{\mathbf{Y}}, E_{\mathbf{X}}, keys_{\mathcal{S}}), keys_{\mathcal{R}}) = \mathbf{X} \cup \mathbf{Y} \tag{23}$$

ii) **Privacy.** The scheme assures privacy of each participant's set if it is secured following Theorem 11 where the definition is instantiated with the PPT functionality

$$f : \mathcal{P}(\mathbb{M}) \times \mathcal{P}(\mathbb{M}) \longrightarrow (\mathcal{P}(\mathbb{M}) \times \mathbb{N}) \times \mathcal{P}(\mathbb{M}) \tag{24}$$

and for inputs $\mathbf{X}, \mathbf{Y} \subset \mathbb{M}$, the ideal output-pair is

$$f(\mathbf{X}, \mathbf{Y}) = ((\mathbf{X} \cup \mathbf{Y}, |\mathbf{Y}|), \emptyset) \tag{25}$$

iii) **Unbalanced efficiency.** For a security parameter κ and sets $\mathbf{X} \subset \mathbb{M}$ for the receiver and $\mathbf{Y} \subset \mathbb{M}$ for the sender, if $|\mathbf{Y}| = o(|\mathbf{X}|)$, then the total communication volume of the scheme, as well as the sender's arithmetic cost, are $o(|\mathbf{X}|)$.

3.3 Assumptions

According to [Brakerski et al.(2014)], the BGV scheme semantic security relies on the general learning with error (GLWE) assumption, which regroups the learning with error (LWE) assumption from [Regev(2009)] and the ring-LHE (RLWE) assumption from [Lyubashevsky et al.(2013)].

Definition 13. (GLWE) For κ a security parameter, $n = n(\kappa)$ an integer dimension, $\phi_d(X)$ the d^{th} cyclotomic polynomial, with $d = d(\kappa)$, and $p = p(\kappa)$ a prime integer. Let $\mathcal{R} = \frac{\mathbb{Z}[X]}{(\phi_d(X))}$, $\mathcal{R}_p = \frac{\mathcal{R}}{(p)}$ and $\chi = \chi(\kappa)$ a distribution over $\mathcal{R}$. The $\text{GLWE}_{n,\phi_d,q,\chi}$ problem is to distinguish the distribution of $(\mathbf{a}_i, b_i)$ taken uniformly at random in $\mathcal{R}_p^n \times \mathcal{R}_p$ from the distribution of $(\mathbf{a}_i, b_i) \in \mathcal{R}_p^n \times \mathcal{R}_p$, where $\mathbf{a}_i$ and $\mathbf{s}$ are taken uniformly at random in $\mathcal{R}_p^n$, e_i is taken following the distribution χ and $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i$. The $\text{GLWE}_{n,\phi_d,q,\chi}$ assumption is that the $\text{GLWE}_{n,\phi_d,q,\chi}$ problem is infeasible.

4 Homomorphic UPSU protocol

The idea of our protocol is to represent each set, the receiver's and the sender's, with polynomials as in [Frikken(2007)]. We use the euclidean remainder to reduce the receiver's polynomial to the size of the sender's polynomial, and we use efficient multipoint evaluation to alleviate the computational cost of the sender. We perform all those operations under homomorphic schemes to keep the sets private, and with some masking and blending, we obtain our UPSU protocol.

Remark 14. In the following protocol, we are making the assumption that a LHE and a FHE can share the same plaintext space. We will see in Section 5 that it is obviously true if we are using a FHE scheme for the entire protocol.

Formally, our protocol is built with the algorithms **Setup**, **Y.Enc**, **X.ExtractEnc**, **Comput** and **Union** respectively presented in Algs. 1 to 5. A more visual version is presented in Protocol 1.

Proposition 15. *The protocol built with the algorithms **Setup**, **Y.Enc**, **X.ExtractEnc**, **Comput** and **Union**, respectively presented in Algs. 1 to 5, is correct.*

Proof. **Correctness.** We assume the correctness of the encryption schemes. Let κ be a security parameter, let $\mathbf{X}$ be the receiver's set and $\mathbf{Y}$ be the sender's set. Let $keys_{\mathcal{R}}$ and $keys_{\mathcal{S}}$ be the outputs of **Setup** (κ). We

Algorithm 1 Setup(κ)

Input: A security parameter κ.
Output: A pair of LHE keys $(pk_{\mathcal{R}}, sk_{\mathcal{R}})$ and a FHE public key $pk_{\mathcal{S}}$.
Output: A pair of FHE keys $(pk_{\mathcal{S}}, sk_{\mathcal{S}})$ and a LHE public key $pk_{\mathcal{R}}$.
Remark: $pk_{\mathcal{R}}$ and $pk_{\mathcal{S}}$ implicitly define the same plaintext space $\mathbb{M}$ but potentially different ciphertext spaces, resp. $\mathbb{E}_L$ and $\mathbb{E}_F$.

1: $\mathcal{R}$: compute $(pk_{\mathcal{R}}, sk_{\mathcal{R}}) \leftarrow$ **L.Setup**(κ) and send $pk_{\mathcal{R}}$ to $\mathcal{S}$;
2: $\mathcal{S}$: compute $(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow$ **F.Setup**(κ) and send $pk_{\mathcal{S}}$ to $\mathcal{R}$;
3: $\mathcal{R}$: **return** $keys_{\mathcal{R}} \leftarrow \{(pk_{\mathcal{R}}, sk_{\mathcal{R}}), pk_{\mathcal{S}}\}$;
4: $\mathcal{S}$: **return** $keys_{\mathcal{S}} \leftarrow \{(pk_{\mathcal{S}}, sk_{\mathcal{S}}), pk_{\mathcal{R}}\}$;

Algorithm 2 Y.Enc($\mathbf{Y}, keys_{\mathcal{S}}$)

Input: A set of plaintext $\mathbf{Y} \subset \mathbb{M}$ and $keys_{\mathcal{S}} = \{(pk_{\mathcal{S}}, sk_{\mathcal{S}}), pk_{\mathcal{R}}\}$.
Output: An encrypted polynomial $\widetilde{P_{\mathcal{S}}} \in \mathbb{E}_F[T]$.

1: $\mathcal{S}$: compute $P_{\mathcal{S}} \leftarrow \prod_{y \in \mathbf{Y}} (T - y)$;
2: $\mathcal{S}$: compute $\widetilde{P_{\mathcal{S}}} \leftarrow$ **F.E**$_{pk_{\mathcal{S}}}(P_{\mathcal{S}})$ and send $\widetilde{P_{\mathcal{S}}}$ to $\mathcal{R}$;
3: $\mathcal{R}$: **return** $\widetilde{P_{\mathcal{S}}}$;

are using the notations of the algorithms. Let $(\widehat{c_i}, \widehat{c_i}')$ be an element of $\widehat{E} \leftarrow$ **Comput**$(\mathbf{Y}, \{\widetilde{R}, \widehat{H}\}, keys_{\mathcal{S}})$ and let (c_i, c_i') be its decrypted tuple with the key $sk_{\mathcal{R}}$. The elements $\{c_i' \times c_i^{-1}\}$ is added to $\mathbf{X}$ if and only if $\widehat{c_i}$ is not an encryption of zero. $\widehat{c_i}$ is an encryption of $r_i(a_i - b_i)$ where r_i is a non-zero random plaintext, b_i is the polynomial R evaluated in y_i and a_i is the polynomial H evaluated in y_i. However, the polynomial R is the sum of H and the euclidean remainder, that we denote S, between $P_{\mathcal{R}}$ and $P_{\mathcal{S}}$ (that is monic). So $a_i - b_i = R(y_i) - H(y_i) = S(y_i)$. It means that $\widehat{c_i}$ is an encryption of zero if and only if y_i is a root of the remainder S and $y_i \in \mathbf{Y}$ is a root of the remainder S if and only if $y_i \in \mathbf{X}$. It is easy to see that, if $c_i \neq 0$, $c_i' \times c_i^{-1} = y_i$. To conclude, the elements added to $\mathbf{X}$ are exactly the elements $y \in \mathbf{Y}$ that are not in $\mathbf{X} \cap \mathbf{Y}$. □

Proposition 16. *The protocol built with the algorithms **Setup**, **Y.Enc**, **X.ExtractEnc**, **Comput** and **Union**, respectively presented in Algs. 1 to 5, is secure under the honest-but-curious adversary model.*

Proof. The complete simulation proof is presented in Appendix A.1. □

Remark 17. In the following, the communication volume counts the number of ciphertext exchanged, and the arithmetic cost denotes the number of basic arithmetic operations needed in the algorithms. Even if basic homomorphic operations (additions, multiplications, encryption and decryption) are more expensive than clear operations, their computational cost is assumed constant. However, we will still distinguish the cost of polynomials products in clear, in LHE and in FHE as the algorithm used may not be the same (we keep using the notations $\mathcal{M}$, $\mathcal{M}_L$ and $\mathcal{M}_F$). We will hide the security parameter in that analysis, considering that it is a constant, but all the costs depend on it.

Proposition 18. *For the receiver owning a set $\mathbf{X}$ of n elements, and the sender owning a set $\mathbf{Y}$ of m elements, with the assumption that $n > m$, the protocol built with the algorithms **Setup**, **Y.Enc**, **X.ExtractEnc**, **Comput** and **Union**, respectively presented in Algs. 1 to 5, computes the set union with the asymptotic complexity bounds presented in Table 2.*

Proof. The **Setup** algorithm is independent of the size on the sets, and only two keys are exchanged. **Y.Enc** consists in the computation of $P_{\mathcal{S}}$ for $\mathcal{S}$, which costs, with a recursive algorithm, $\frac{1}{2}\mathcal{M}(m)\log m$ basic operations; encrypting the polynomial requires m encryptions, so $O(m)$ computations, and sending this

Algorithm 3 $\mathbf{X.ExtractEnc}(\mathbf{X}, keys_{\mathcal{R}}, \widetilde{P_{\mathcal{S}}})$

Input: A set of plaintext $\mathbf{X} \subset \mathbb{M}$, $keys_{\mathcal{R}} = \{(pk_{\mathcal{R}}, sk_{\mathcal{R}}), pk_{\mathcal{S}}\}$ and a ciphertext polynomial $\widetilde{P_{\mathcal{S}}}$.
Output: Two encrypted polynomials $\widetilde{R} \in \mathbb{E}_F[T]$ and $\widehat{H} \in \mathbb{E}_L[T]$

$\mathcal{R}$: compute $P_{\mathcal{R}} \leftarrow \prod_{x \in \mathbf{X}} (T - x)$;
$\mathcal{R}$: compute $\widetilde{P_{\mathcal{R}}} \leftarrow \mathbf{F.E}_{pk_{\mathcal{S}}}(P_{\mathcal{R}})$;
$\mathcal{R}$: randomly select $H \xleftarrow{\$} \mathbb{M}[T]$ such that $\deg(H) = \deg(\widetilde{P_{\mathcal{S}}}) - 1$;
$\mathcal{R}$: compute $\widetilde{H} \leftarrow \mathbf{F.E}_{pk_{\mathcal{S}}}(H)$;
$\mathcal{R}$: compute $\widehat{H} \leftarrow \mathbf{L.E}_{pk_{\mathcal{R}}}(H)$;
$\mathcal{R}$: compute $\widetilde{R} \leftarrow \mathbf{F.Rem}(\widetilde{P_{\mathcal{R}}}, \widetilde{P_{\mathcal{S}}}) +_F \widetilde{H}$;
$\mathcal{R}$: send $\{\widetilde{R}, \widehat{H}\}$ to $\mathcal{S}$;
$\mathcal{S}$: **return** $\{\widetilde{R}, \widehat{H}\}$;

Algorithm 4 $\mathbf{Comput}(\mathbf{Y}, \{\widetilde{R}, \widehat{H}\}, keys_{\mathcal{S}})$

Input: A set of m plaintexts $\mathbf{Y} \subset \mathbb{M}$, two ciphertext polynomials $\{\widetilde{R}, \widehat{H}\} \in \mathbb{E}_F[T] \times \mathbb{E}_L[T]$, and $keys_{\mathcal{S}} = \{(pk_{\mathcal{S}}, sk_{\mathcal{S}}), pk_{\mathcal{R}}\}$.
Output: A set of ciphertext pairs $\widehat{E} \subset \mathbb{E}_L \times \mathbb{E}_L$.

$\mathcal{S}$: compute $R \leftarrow \mathbf{F.D}_{sk_{\mathcal{S}}}(\widetilde{H}) \in \mathbb{M}[T]$;
$\mathcal{S}$: compute $\{a_i\}_{i \in \{1,\dots,m\}} \leftarrow \mathbf{MultEv}(R, \mathbf{Y}) \subset \mathbb{M}$;
$\mathcal{S}$: compute $\{\widehat{b_i}\}_{i \in \{1,\dots,m\}} \leftarrow \mathbf{L.MultEv}(\widehat{H}, \mathbf{Y}) \subset \mathbb{E}_L$;
for all $i \in \{1, \dots, m\}$ **do**
    $\mathcal{S}$: randomly select $r_i \xleftarrow{\$} \mathbb{M} \setminus \{0\}$;
    $\mathcal{S}$: compute $\widehat{a_i} \leftarrow \mathbf{L.E}_{pk_{\mathcal{R}}}(a_i) \in \mathbb{E}_L$;
    $\mathcal{S}$: compute $\widehat{c_i} \leftarrow r_i \ltimes_L (\widehat{a_i} -_L \widehat{b_i}) \in \mathbb{E}_L$, $\widehat{c_i}' \leftarrow y_i \ltimes_L \widehat{c_i} \in \mathbb{E}_L$;
end for
$\mathcal{S}$: randomly select $\pi \xleftarrow{\$} \mathfrak{S}_m$;
$\mathcal{S}$: compute $\widehat{E} \leftarrow \{(\widehat{c_{\pi(i)}}, \widehat{c_{\pi(i)}}')\}_{i \in \{1,\dots,m\}} \subset \mathbb{E}_L \times \mathbb{E}_L$;
$\mathcal{S}$: send $\widehat{E}$ to $\mathcal{R}$;
$\mathcal{R}$: **return** $\widehat{E}$;

polynomial to $\mathcal{R}$ is equivalent to send m ciphertexts. In **X.ExtractEnc**, $\mathcal{R}$ computes $P_{\mathcal{R}}$ in $\frac{1}{2}\mathcal{M}(n)\log n$, and encrypts it in n computations; then, the homomorphic remainder needs $\frac{9}{2}\mathcal{M}_F(n-m) + O(n-m)$ arithmetic operation as said in Theorem 10; the polynomial encryptions of H and addition with H are in $O(m)$ and the polynomials sent have both degrees $m-1$, so the communication volume is in $O(m)$. **Comput** requires m encryptions and decryptions, m homomorphic additions and clear/ciphered products, which are all in $O(m)$; $\mathcal{S}$ has to perform a multipoint evaluation both in plaintext and in ciphertext on polynomials of degrees $m-1$ in its m elements; the precomputation told in Theorem 8 is already done when computing $P_{\mathcal{S}}$ so it adds $\mathcal{M}_L(m)\log m + \mathcal{M}(m)\log m + O(m\log m)$ to the cost; then m pairs of ciphertexts are sent, so a communication volume in $O(m)$. Finally, **Union** requires at most $2m$ decryptions and m products and inversions, so $O(m)$ computations. □

Overall, we have shown in Theorem 19 that the protocol summarized in Protocol 1 is a secure unbalanced private set union scheme

Theorem 19. *The protocol built with the algorithms **Setup**, **Y.Enc**, **X.ExtractEnc**, **Comput** and **Union**, respectively presented in Algs. 1 to 5, is a secure unbalanced private set union scheme (UPSU) under the honest-but-curious adversary model.*

Algorithm 5 Union$(\mathbf{X}, \widehat{E}, keys_{\mathcal{R}})$

Input: A set of plaintexts $\mathbf{X} \subset \mathbb{M}$, a set of ciphertext pairs $\widehat{E} \subset \mathbb{E}_L \times \mathbb{E}_L$ and $keys_{\mathcal{R}} = \{(pk_{\mathcal{R}}, sk_{\mathcal{R}}), pk_{\mathcal{S}}\}$.
Output: The union set $\mathbf{X}$.

for $(\widehat{c_i}, \widehat{c_i}') \in \widehat{E}$ **do**
  $\mathcal{R}$: compute $c_i \leftarrow \mathbf{L.D}_{sk_{\mathcal{R}}}(\widehat{c_i}) \in \mathbb{M}$;
  if $c_i \neq 0$ **then**
    $\mathcal{R}$: compute $c_i' \leftarrow \mathbf{L.D}_{sk_{\mathcal{R}}}(\widehat{c_i}') \in \mathbb{M}$;
    $\mathcal{R}$: compute $\mathbf{X} \leftarrow \mathbf{X} \cup \{c_i' \times c_i^{-1}\}$;
  end if
end for
$\mathcal{R}$: **return** $\mathbf{X}$;

Table 2: Cost analysis of Protocol 1

Algorithm	Ar. Cost for $\mathcal{R}$	Ar. Cost for $\mathcal{S}$	Comm. Vol.
Setup	$O(1)$	$O(1)$	$O(1)$
Y.Enc	$O(1)$	$\frac{1}{2}\mathcal{M}(m)\log m$	$O(m)$
X.ExtractEnc	$\frac{9}{2}\mathcal{M}_F(n-m)+$ $\frac{1}{2}\mathcal{M}(n)\log n\ +$ $O(n-m)$	$O(1)$	$O(m)$
Comput	$O(1)$	$\mathcal{M}_L(m)\log m+$ $\mathcal{M}(m)\log m\ +$ $O(m\log m)$	$O(m)$
Union	$O(m)$	$O(1)$	$O(1)$

5 Instantiation of LHE and FHE with BGV

The direct way to instantiate our UPSU Protocol 1 is to use a fully homomorphic encryption scheme for the entire protocol. It avoids any possible conflict of compatibility between the plaintext spaces of LHE and FHE. One of the most efficient FHE encryption scheme available nowadays is the BGV cryptosystem presented in [Brakerski et al.(2014)], whose security is based on the GLWE assumption. Thanks to Shoup and Halevi, we can use an implementation of the bootstrappable scheme in the C++ open source library HElib[2] [Halevi and Shoup(2014), Halevi and Shoup(2021)]. Another implementation of the BGV scheme is available in the open-source library Microsoft SEAL[3]. There is also an active research to increase the efficiency of the RLWE based schemes, in particular to speedup the bootstrapping procedure [Geelen and Vercauteren(2023), Guimarães et al.(2023)]. For now, the computation of homomorphic multiplication and the bootstraping procedure are usually quite slow for bootstrappable contexts with decent security. As the research is active on the subject, we expect to have a better efficiency in the future years. In the following, we will briefly present the BGV cryptosystem and compare our estimated client (sender) time and communication volume for Protocol 1 to the values presented in [Tu et al.(2023), Table 3].

5.1 BGV cryptosystem

We here give an overview of the BGV cryptosystem, and we refer the reader to [Brakerski et al.(2014)] for more details. First, the plaintext space is a ring $\mathcal{R}_{p^r} = \frac{\mathbb{Z}[X]}{(p^r, \phi_d(X))}$ where p is a prime number, and $\phi_d(X)$ is the d^{th} cyclotomic polynomial. In this paper, we will always consider $r = 1$. The ciphertext space is $\mathcal{R}_q^2$ where $\mathcal{R}_q = \frac{\mathbb{Z}[X]}{(q, \phi_d(X))}$ and q is an odd modulus that might change during computation. The secret key is

[2] https://github.com/homenc/HElib
[3] https://github.com/microsoft/SEAL

Protocol 1: Communication optimal UPSU Protocol

	$\mathcal{R}$		$\mathcal{S}$	
	$\mathbf{X} = \{x_1, ..., x_n\} \subset \mathbb{M}$		$\mathbf{Y} = \{y_1, ..., y_m\} \subset \mathbb{M}$	
Setup	$\{(pk_{\mathcal{R}}, sk_{\mathcal{R}}) \leftarrow \mathbf{L.Setup}(\kappa), pk_{\mathcal{S}}\}$		$\{(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow \mathbf{F.Setup}(\kappa), pk_{\mathcal{R}}\}$	**Setup**
	$P_{\mathcal{R}} \leftarrow \prod_{x_i \in \mathbf{X}} (T - x_i)$		$P_{\mathcal{S}} \leftarrow \prod_{y_i \in \mathbf{Y}} (T - y_i)$	
	$\widetilde{P_{\mathcal{R}}} \leftarrow \mathbf{F.E}_{pk_{\mathcal{S}}}(P_{\mathcal{R}})$	$\xleftarrow{\widetilde{P_{\mathcal{S}}}}$	$\widetilde{P_{\mathcal{S}}} \leftarrow \mathbf{F.E}_{pk_{\mathcal{S}}}(P_{\mathcal{S}})$	**Y.Enc**
	$H \xleftarrow{\$} \mathbb{M}[T]_{m-1}$ $\widehat{H} \leftarrow \mathbf{L.E}_{pk_{\mathcal{R}}}(H)$ $\widetilde{H} \leftarrow \mathbf{F.E}_{pk_{\mathcal{S}}}(H)$			
X.ExtractEnc	$\widetilde{R} \leftarrow \mathbf{F.Rem}(\widetilde{P_{\mathcal{R}}}, \widetilde{P_{\mathcal{S}}}) +_F \widetilde{H}$	$\xrightarrow{\widetilde{R},\ \widehat{H}}$	$R \leftarrow \mathbf{F.D}_{sk_{\mathcal{S}}}(\widetilde{R})$ $\{a_i\}_{i\in[m]} \leftarrow \mathbf{MultEv}(R, \mathbf{Y})$ $\{\widehat{b_i}\}_{i\in[m]} \leftarrow \mathbf{L.MultEv}(\widehat{H}, \mathbf{Y})$ $\forall i \in \{1, ..., m\}:$ $r_i \xleftarrow{\$} \mathbb{M} \setminus \{0\}$ $\widehat{a_i} \leftarrow \mathbf{L.E}_{pk_{\mathcal{R}}}(a_i)$ $\widehat{e_i} \leftarrow r_i \ltimes_L (\widehat{a_i} -_L \widehat{b_i})$ $\widehat{e_i}' \leftarrow y_i \ltimes_L \widehat{e_i}$ $\pi \xleftarrow{\$} \mathfrak{S}_m$	
	$\forall(\widehat{e_i}, \widehat{e_i'}) \in \widehat{E}$ $e_i \leftarrow \mathbf{L.D}_{sk_{\mathcal{R}}}(\widehat{e_i})$ $If\ e_i \neq 0:$ $e_i' \leftarrow \mathbf{L.D}_{sk_{\mathcal{R}}}(\widehat{e_i'})$	$\xleftarrow{\widehat{E}}$	$\widehat{E} \leftarrow \{(\widehat{e_{\pi(i)}}, \widehat{e_{\pi(i)}}')\}$	**Comput**
Union	$\mathbf{X} \leftarrow \mathbf{X} \cup \{e_i' \times e_i^{-1}\}$			
	Return X			

a vector $(1, \mathbf{s}) \in \mathcal{R}_q^2$ where $\mathbf{s}$ has a small coefficients (usually in $\{-1, 0, 1\}$). The public key is basically a two entries matrix A where the first entry is a multiple of the secret key masked with an error term and the second is the opposite of the "multiple" term. An encryption of a plaintext $m \in \mathcal{R}_p$ is then a pair (c_0, c_1) such that $c_0 + c_1\mathbf{s} = m + pe \mod q$, where e is a "small" error term. The decryption of a ciphertext (c_0, c_1) is done with the inner product and the modulus reduction $[\langle(1, \mathbf{s}), (c_0, c_1)\rangle]_p$.

Having two ciphertexts (c_0, c_1) and (c_0', c_1') such that $c_0 + c_1\mathbf{s} = m + pe \mod q$ and $c_0' + c_1'\mathbf{s} = m' + pe' \mod q$, we can see that the ciphertext $(c_0'', c_1'') := (c_0 + c_0', c_1 + c_1')$ satisfies $c_0'' + c_1''\mathbf{s} = m + m' + pe'' \mod q$ so can be decrypted to $m + m'$ if the noise e'' has a small enough norm. For the same two ciphertexts, if we consider the vector $(c_0'', c_1'', c_2'') = (c_0c_0', c_0c_1' + c_1c_0', c_1c_1') = (c_0, c_1) \otimes (c_0', c_1')$, we see that $c_0'' + c_1''\mathbf{s} + c_2''\mathbf{s}^2 = mm' + pe'' \mod q$, with $e'' = mpe' + m'pe + ee'p^2$. With the help of a procedure called the key switching procedure, one is able to turn the ciphertext (c_0'', c_1'', c_2''), that can be seen as an encryption under the key $(1, \mathbf{s}, \mathbf{s}^2)$, to a ciphertext $(\overline{c_0}, \overline{c_1})$ such that $\overline{c_0} + \mathbf{s}\overline{c_1} = mm' + pe'' \mod q$. One more time, if the norm of the noise e'' is small enough, this ciphertext can be decrypted to the product mm'. We saw that we can perform homomorphic operations with that scheme as long as the noise is controlled, but those operations, in particular the multiplication, add some noise, so this scheme requires a noise management. The first procedure to manage the noise is the modulus switching, which will reduce the modulus q of the ciphertext space and the noise norm proportionally. This first procedure makes this scheme a leveled homomorphic encryption scheme, because after some several such switches, the modulus cannot be reduced anymore. The second procedure, called bootstraping, allows a "reset" of the noise. But this comes with an increase of the ciphertext modulus via an homomorphic decryption. This procedure is more costly, but allows the scheme to be fully homomorphic.

5.2 Experiments

All our experiments were run a single core of an i7-6700 CPU 3.40GHz.

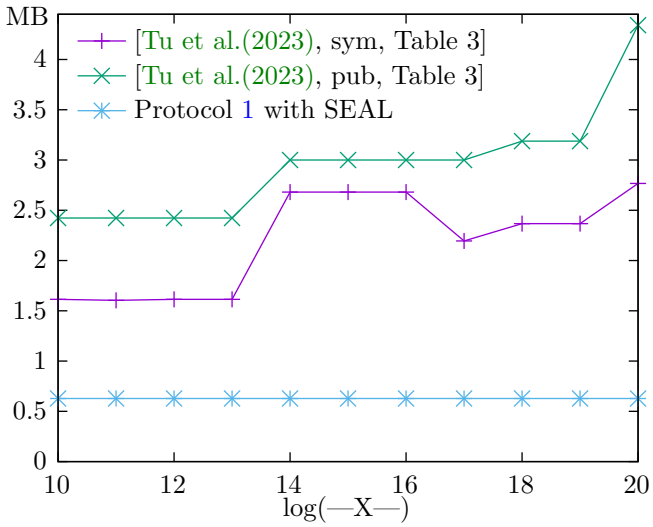
The main focus of this paper is a protocol with optimal asymptotic communication volume. In this protocol, the sender computational cost is also independent from the size of the receiver's large set. In this section we provide preliminary estimates on the different practical costs

To test the receiver side, we need bootstrappable fully homomorphic modular operations and, up to our knowledge, the HElib library is among the only few that provide this. Unfortunately, for instance, in a bootstrappable context with equivalent security $\kappa = 100$, cyclotomic polynomial $\phi_d(X)$ for $d = 46235$ and plaintext modulus $p = 17$, a single ciphertext multiplication requires about 0.45 seconds and a single bootstrap, about 8 minutes. Therefore, in order to run the receiver side of our protocol it seems that for now multi-core servers are recommended. Further work is needed to reduce this in practice. It also might be possible to use batching for the polynomial division.

On the communication and sender sides, however, the situation is more favorable. A trade-off has to be made between fast routines for multi-point evaluation (that have a ciphertext-cleartext multiplicative depth that can be $O(\log(m))$ and naive routines (quadratic time, but of multiplicative depth 1). We here report preliminary results with a naive multi-point evaluation. In order to be able to compare our communication volume with state of the art implementations, we switch to the SEAL library for these tests.

With a computational security $\kappa = 123.1$, and a sender owning a set $\mathbf{Y}$ of $m = |Y| = 2^{10}$ items of 128 bits, SEAL generates a context with a 20-bits prime, $p = 1032193$, and can encrypt the m items in two ciphertext matrices, each one containing 4096 slots. For these parameters, SEAL reports that the serialization of one matrix will require 526 449 bits, and as our Protocol 1 requires to exchange 5 times m ciphertexts ($\widetilde{P_{\mathcal{S}}}$, $\widetilde{R}$, $\widehat{H}$ and $\widehat{E}$, the latter containing $2m$ ciphertexts), the total communication volume is contained in 10 such ciphertext matrices. This represents 0.628MB overall. Figure 1 compares this with [Tu et al.(2023), Table 3] where the memory footprint depends also on $n = |X|$.

Figure 1: Estimated Memory footprint for $m = |Y| = 2^{10}$



Finally, we provide in Table 3 the associated timings for the sender, with $m = |Y| = 2^{10}$, for any $n = |X|$. In this table, `Decryp.` is for the decryption of the masked remainder, `Clear Eval` is for the clear multi-point evaluation of this masked remainder, `Hom. Eval` is for the multi-point evaluation of the ciphered mask, `Filter` is for the remaining arithmetic operations of the sender, while `Total` is the total sender time.

Table 3: Protocol 1 with SEAL, sender time (seconds)

$m = \|Y\|$	Decryp.	Clear Eval.	Hom. Eval.	Filter.	Total
2	0.002	<0.001	0.006	0.012	0.021
4	0.002	<0.001	0.013	0.024	0.039
8	0.002	<0.001	0.025	0.048	0.076
16	0.002	<0.001	0.049	0.096	0.147
32	0.002	<0.001	0.097	0.191	0.290
64	0.002	<0.001	0.193	0.379	0.574
128	0.002	<0.001	0.977	0.756	1.735
256	0.002	0.001	0.756	1.505	2.263
512	0.002	0.003	1.498	2.997	4.499
1024	0.002	0.012	2.980	5.988	8.982
2048	0.002	0.047	5.954	11.964	17.966
4096	0.002	0.186	11.964	23.938	36.090

In this benchmark we are using a naive algorithm for the multi-point evaluations. These two steps require a quadratic number of operations, while the other two are linear. The cleartext multi-point evaluation is performed fully in clear and is thus much faster than the rest. With the large dimensions, the quadratic behavior starts to show. Then, each homomorphic polynomial evaluation is batched with 4096 slots. Therefore the quadratic behavior does not show in this column: at each evaluation point a single batch is sufficient. Thus, overall, the homomorphic operations are largly dominating. Further, if the sender's set remains in the range of Table 3, its computational effort remains under a minute while exhibiting a linear behavior.

References

[Badrinarayanan et al.(2022)] Saikrishna Badrinarayanan, Peihan Miao, and Tiancheng Xie. 2022. Updatable Private Set Intersection. *Proc. Priv. Enhancing Technol.* 2022, 2 (2022), 378–406. https://doi.org/10.2478/POPETS-2022-0051

[Bostan et al.(2003)] Alin Bostan, Grégoire Lecerf, and Éric Schost. 2003. Tellegen's principle into practice. In *Symbolic and Algebraic Computation, International Symposium ISSAC 2003, Drexel University, Philadelphia, Pennsylvania, USA, August 3-6, 2003, Proceedings*, J. Rafael Sendra (Ed.). ACM, 37–44. https://doi.org/10.1145/860854.860870

[Brakerski et al.(2014)] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory* 6, 3 (2014), 13:1–13:36. https://doi.org/10.1145/2633600

[Brickell and Shmatikov(2005)] Justin Brickell and Vitaly Shmatikov. 2005. Privacy-Preserving Graph Algorithms in the Semi-honest Model. In *Advances in Cryptology - ASIACRYPT 2005*, Bimal Roy (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 236–252.

[Davidson and Cid(2017)] Alex Davidson and Carlos Cid. 2017. An Efficient Toolkit for Computing Private Set Operations. In *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 10343)*, Josef Pieprzyk and Suriadi Suriadi (Eds.). Springer, 261–278. https://doi.org/10.1007/978-3-319-59870-3_15

[Frikken(2007)] Keith B. Frikken. 2007. Privacy-Preserving Set Union. In *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4521)*, Jonathan Katz and Moti Yung (Eds.). Springer, 237–252. https://doi.org/10.1007/978-3-540-72738-5_16

[Garimella et al.(2021)] Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. 2021. Private Set Operations from Oblivious Switching. In *Public-Key Cryptography - PKC 2021 -*

24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12711), Juan A. Garay (Ed.). Springer, 591–617. https://doi.org/10.1007/978-3-030-75248-4_21

[Geelen and Vercauteren(2023)] Robin Geelen and Frederik Vercauteren. 2023. Bootstrapping for BGV and BFV Revisited. *J. Cryptol.* 36, 2 (2023), 12. https://doi.org/10.1007/S00145-023-09454-6

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

[Groce et al.(2019)] Adam Groce, Peter Rindal, and Mike Rosulek. 2019. Cheaper Private Set Intersection via Differentially Private Leakage. *Proc. Priv. Enhancing Technol.* 2019, 3 (2019), 6–25. https://doi.org/10.2478/POPETS-2019-0034

[Guimarães et al.(2023)] Antonio Guimarães, Hilder V. L. Pereira, and Barry Van Leeuwen. 2023. Amortized Bootstrapping Revisited: Simpler, Asymptotically-faster, Implemented. *IACR Cryptol. ePrint Arch.* 2023, 14 (2023), 14. https://eprint.iacr.org/2023/014

[Halevi and Shoup(2014)] Shai Halevi and Victor Shoup. 2014. Algorithms in HElib. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 8616)*, Juan A. Garay and Rosario Gennaro (Eds.). Springer, 554–571. https://doi.org/10.1007/978-3-662-44371-2_31

[Halevi and Shoup(2021)] Shai Halevi and Victor Shoup. 2021. Bootstrapping for HElib. *J. Cryptol.* 34, 1 (2021), 7. https://doi.org/10.1007/s00145-020-09368-7

[Jia et al.(2022)] Yanxue Jia, Shi-Feng Sun, Hong-Sheng Zhou, Jiajun Du, and Dawu Gu. 2022. Shuffle-based Private Set Union: Faster and More Secure. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 2947–2964. https://www.usenix.org/conference/usenixsecurity22/presentation/jia

[Kiss et al.(2017)] Ágnes Kiss, Jian Liu, Thomas Schneider, N. Asokan, and Benny Pinkas. 2017. Private Set Intersection for Unequal Set Sizes with Mobile Applications. *Proc. Priv. Enhancing Technol.* 2017, 4 (2017), 177–197. https://doi.org/10.1515/POPETS-2017-0044

[Kissner and Song(2005)] Lea Kissner and Dawn Song. 2005. Privacy-Preserving Set Operations. In *Advances in Cryptology – CRYPTO 2005*, Victor Shoup (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 241–257.

[Kolesnikov et al.(2019)] Vladimir Kolesnikov, Mike Rosulek, Ni Trieu, and Xiao Wang. 2019. Scalable Private Set Union from Symmetric-Key Techniques. In *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11922)*, Steven D. Galbraith and Shiho Moriai (Eds.). Springer, 636–666. https://doi.org/10.1007/978-3-030-34621-8_23

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

[Lyubashevsky et al.(2013)] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013. On Ideal Lattices and Learning with Errors over Rings. *J. ACM* 60, 6 (2013), 43:1–43:35. https://doi.org/10.1145/2535925

[Morales et al.(2023)] Daniel Morales, Isaac Agudo, and Javier Lopez. 2023. Private set intersection: A systematic literature review. *Computer Science Review* 49 (2023), 100567. https://doi.org/10.1016/j.cosrev.2023.100567

[Ramanathan et al.(2020)] Sivaramakrishnan Ramanathan, Jelena Mirkovic, and Minlan Yu. 2020. BLAG: Improving the Accuracy of Blacklists. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society. https://www.ndss-symposium.org/ndss-paper/blag-improving-the-accuracy-of-blacklists/

[Regev(2009)] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6 (2009), 34:1–34:40. https://doi.org/10.1145/1568318.1568324

[Resende and de Freitas Aranha(2021)] Amanda Cristina Davi Resende and Diego de Freitas Aranha. 2021. Faster unbalanced Private Set Intersection in the semi-honest setting. *J. Cryptogr. Eng.* 11, 1 (2021), 21–38. https://doi.org/10.1007/S13389-020-00242-7

[Tu et al.(2023)] Binbin Tu, Yu Chen, Qi Liu, and Cong Zhang. 2023. Fast Unbalanced Private Set Union from Fully Homomorphic Encryption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda (Eds.). ACM, 2959–2973. https://doi.org/10.1145/3576915.3623064

[Zhang et al.(2023)] Cong Zhang, Yu Chen, Weiran Liu, Min Zhang, and Dongdai Lin. 2023. Linear Private Set Union from Multi-Query Reverse Private Membership Test. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, 337–354. https://www.usenix.org/conference/usenixsecurity23/presentation/zhang-cong

A Security Proofs

A.1 Security Proof for Protocol 1

We assume that both the FHE and LHE schemes used are semantically secure. In the following, we denote the receiver $\mathcal{R}$ as the party 1 and the sender $\mathcal{S}$ as the party 2 and Protocol 1 will be called Π. This protocol has 3 rounds: $\mathcal{R}$ receives 2 messages M_1 and M_3 while $\mathcal{S}$ receives only M_2. The semantic functionality is $f : \mathcal{P}(\mathbb{M}) \times \mathcal{P}(\mathbb{M}) \to (\mathcal{P}(\mathbb{M}) \times \mathbb{N}) \times \mathcal{P}(\mathbb{M})$ where $\mathcal{P}$ denotes the power set. The ideal output-pair is $f(\mathbf{X}, \mathbf{Y}) = (f_1(\mathbf{X}, \mathbf{Y}), f_2(\mathbf{X}, \mathbf{Y})) = ((\mathbf{X} \cup \mathbf{Y}, |\mathbf{Y}|), \emptyset)$. The following views are reduced to the minimal set that could trivially imply the real view; for example, if the real view have a clear polynomial R, a key pk and a ciphertext $\mathbf{F.E}_{pk}(R)$, we omit $\mathbf{F.E}_{pk}(R)$ in the view, because if we can simulate both R and pk, it is trivial to simulate $\mathbf{F.E}_{pk}(R)$. The views and the outputs of each parties are:

- $\mathbf{view}_1^{\Pi}(\mathbf{X}, \mathbf{Y}) = (\mathbf{X}, C_1, M_1, M_3)$ where:

$$C_1 = \Big\{ \widetilde{S}, H, pk_{\mathcal{R}}, sk_{\mathcal{R}}, pk_{\mathcal{S}}, \\ \left\{ \left(r_{\pi(i)} S(y_{\pi(i)}),\ y_{\pi(i)} r_{\pi(i)} S(y_{\pi(i)}) \right) \right\}_{1 \le i \le |\mathbf{Y}|} \Big\}$$

  Where $\widetilde{S} \leftarrow \mathbf{F.Rem}(\widetilde{P_{\mathcal{R}}}, \widetilde{P_{\mathcal{S}}})$ and S is its decryption with the key $sk_{\mathcal{S}}$. The content of the messages are:

$$M_1 = \left\{ \widetilde{P_{\mathcal{S}}} \right\}$$
$$M_3 = \left\{ \left\{ \left(\mathbf{L.E}_{pk_{\mathcal{R}}} \left(r_{\pi(i)} S(y_{\pi(i)}) \right), \mathbf{L.E}_{pk_{\mathcal{R}}} \left(y_{\pi(i)} r_{\pi(i)} S(y_{\pi(i)}) \right) \right) \right\}_{1 \le i \le |\mathbf{Y}|} \right\}$$

- $\mathbf{view}_2^{\Pi}(\mathbf{X},\mathbf{Y}) = (\mathbf{Y}, C_2, M_2)$ where:

$$C_2 = \left\{pk_{\mathcal{S}}, sk_{\mathcal{S}}, pk_{\mathcal{R}}, R, \{r_i\}_{1\leq i\leq |\mathbf{Y}|}, \pi\right\}$$
$$M_2 = \left\{\widetilde{R}, \widehat{H}\right\}$$

- $\mathbf{output}_1^{\Pi}(\mathbf{X},\mathbf{Y}) = \left(\mathbf{X} \cup \left\{y_{\pi(i)} | S(y_{\pi(i)}) \neq 0\right\}, |\mathbf{Y}|\right)$
- $\mathbf{output}_2^{\Pi}(\mathbf{X},\mathbf{Y}) = \emptyset$

On the side of $\mathcal{S}$, a probabilistic polynomial-time algorithm S_2, taking as input the set $\mathbf{Y}$, should simulate $\mathbf{view}_2^{\Pi}(\mathbf{X},\mathbf{Y})$ with the following tuple.

$$S_2(\mathbf{Y},\emptyset) = (\mathbf{Y}, \left\{pk, sk, pk', R_1, \{r_i'\}_{1\leq i\leq |\mathbf{Y}|}, \pi'\right\}, \\ \{\mathbf{F.E}_{pk}(R_1), \mathbf{L.E}_{pk'}(R_2)\})$$

Where $(sk, pk) \leftarrow \mathbf{F.Setup}(\kappa)$, $pk' \in \mathbf{L.Setup}(\kappa)$, R_1 and R_2 are random polynomials in $\mathbb{M}[T]$ of degrees $|\mathbf{Y}|-1$, $\{r_i'\}_{1\leq i\leq |\mathbf{Y}|}$ is a set of random values in $\mathbb{M}\setminus\{0\}$ and finally π' randomly selected in $\mathfrak{S}_{|\mathbf{Y}|}$.
As in the protocol S is of degree at most $|\mathbf{Y}|-1$ and H is taken uniformly at random of degree $|\mathbf{Y}|-1$, one cannot distinguish $R = S + H$ from R_1, for R_1 taken uniformly at random of size $|\mathbf{Y}|-1$. This implies also that $\mathbf{F.E}_{pk}(R_1)$ is a good simulation of $\widetilde{R}$. The set of r_i' obviously simulates well the set of r_i as those are both taken as random non-zero plaintexts. As the encryption schemes are assumed semantically secure, $\widehat{H}$ is indistinguishable from $\mathbf{L.E}_{pk'}(R_2)$, if R_2 is taken randomly of same degree than H. Finally, we obtain for every subsets $\mathbf{X},\mathbf{Y} \subset \mathbb{M}$:

$$\{S_2(\mathbf{Y},\emptyset), ((\mathbf{X}\cup\mathbf{Y}, |\mathbf{Y}|), \emptyset)\} \stackrel{c}{\equiv} \{\mathbf{view}_2^{\Pi}(\mathbf{X},\mathbf{Y}), \mathbf{output}^{\Pi}(\mathbf{X},\mathbf{Y})\}$$

On the side of $\mathcal{R}$, a probabilistic polynomial-time algorithm S_1 taking as input the set $\mathbf{X}$ and $(\mathbf{X}\cup\mathbf{Y}, |\mathbf{Y}|)$ should simulate $\mathbf{view}_1^{\Pi}(\mathbf{X},\mathbf{Y})$ this way:

$$S_1(\mathbf{X}, (\mathbf{X}\cup\mathbf{Y}, |\mathbf{Y}|)) = \Bigg(\mathbf{X}, \\ \left\{R_3, R_2, pk, sk, pk', \{(r_i^{(1)}, r_i^{(1)}r_i^{(2)})\}_{1\leq i\leq |\mathbf{Y}|}\right\}, \\ \{\mathbf{F.E}_{pk'}(R_1)\}, \\ \left\{\{(\mathbf{L.E}_{pk}(r_i^{(1)}), \mathbf{L.E}_{pk}(r_i^{(1)}r_i^{(2)}))\}_{1\leq i\leq |\mathbf{Y}|}\right\}\Bigg)$$

Let $\delta \leftarrow |\mathbf{Y}| - (|\mathbf{X}\cup\mathbf{Y}| - |\mathbf{X}|)$: this is the size of the intersection $\mathbf{X}\cap\mathbf{Y}$. Let $\{m_1, ..., m_\delta\}$ be δ distinct random values taken in $\mathbf{X}$. Then, let $R_1 \in \mathbb{M}[T]$ be the product $\prod_{i=1}^{\delta}(T-m_i) \prod_{m\in\mathbf{X}\cup\mathbf{Y}\setminus\mathbf{X}}(T-m)$, R_2 be a random polynomial in $\mathbb{M}[T]$ of degree $|\mathbf{Y}|-1$, and $R_3 \leftarrow \mathbf{F.Rem}_{pk'}(\widetilde{P_{\mathcal{R}}}, \mathbf{F.E}_{pk'}(R_1))$, for $(., pk') \leftarrow \mathbf{F.Setup}(\kappa)$. $(sk, pk) \leftarrow \mathbf{El.Setup}(\kappa)$. Also let $|\mathbf{X}\cup\mathbf{Y}| - |\mathbf{X}|$ of the $r_i^{(1)}$ be random plaintexts in $\mathbb{M}\setminus\{0\}$ and set the δ others to zero with the indices of the zeroes uniformly distributed. If index i is such that $r_i^{(1)} = 0$, then $r_i^{(2)}$ is taken to be equal to zero too, and for the indices i such that $r_i^{(1)} \neq 0$, then $r_i^{(2)}$ is randomly taken in $\mathbf{X}\cup\mathbf{Y}\setminus\mathbf{X}$, with each element in $\mathbf{X}\cup\mathbf{Y}\setminus\mathbf{X}$ selected once and only once. From the semantic security of the encryption schemes and the indistinguishability of keys, $\mathbf{F.E}_{pk'}(R_1)$ is indistinguishable from $\widetilde{P_{\mathcal{S}}}$. Therefore, this implies that R_3 is indistinguishable from $\widetilde{S}$. Now, H is randomly selected in $\mathbb{M}[T]$ of degree $|\mathbf{Y}|-1$ in the protocol, it is thus well simulated by R_2. The set $\{(r_i^{(1)}, r_i^{(1)}r_i^{(2)})\}_{1\leq i\leq |\mathbf{Y}|}$ contains δ zeroes, and $|\mathbf{Y}| - \delta$ random pairs, such that the division of the second element of the tuple by the first one gives an element added to $\mathbf{X}$ by the protocol: this is thus indistinguishable from $\left\{\left(r_{\pi(i)}S(y_{\pi(i)}), y_{\pi(i)}r_{\pi(i)}S(y_{\pi(i)})\right)\right\}_{1\leq i\leq |\mathbf{Y}|}$. Overall we have that an encryption of the first set of pairs under a LHE scheme is a good simulation of the encryption of the second set of pairs under a LHE.

To conclude, we obtain for every subsets $\mathbf{X}, \mathbf{Y} \subset \mathcal{M}$:

$$\{S_1(\mathbf{X}, (\mathbf{X} \cup \mathbf{Y}, |\mathbf{Y}|)), ((\mathbf{X} \cup \mathbf{Y}, |\mathbf{Y}|), \emptyset)\} \stackrel{c}{\equiv} \{\mathbf{view}_1^{\Pi}(\mathbf{X}, \mathbf{Y}), \mathbf{output}^{\Pi}(\mathbf{X}, \mathbf{Y})\}$$