Communication Optimal Unbalanced Private Set Union

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

▶ **To cite this version:**

Jean-Guillaume Dumas, Alexis Galan, Bruno Grenet, Aude Maignan, Daniel S. Roche. Communication Optimal Unbalanced Private Set Union. 2024. hal-04475604v3

HAL Id: hal-04475604
https://hal.science/hal-04475604v3

Preprint submitted on 8 Jul 2024 (v3), last revised 2 Oct 2024 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Communication Optimal Unbalanced Private Set Union

Extended abstract

Jean-Guillaume Dumas*[1], Alexis Galan†[1], Bruno Grenet‡[1], Aude 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 a protocol allowing a whistleblower to report an alert to an institution. The institution is expected to have much more computational resources than the whistleblower, who might only use a phone to communicate. The expected security in such a protocol is the following: The institution should not learn which data are common to its data set and the whistleblower's, because it could leak some clue on the identity of the whistleblower and compromise its anonymity. Also, the whistleblower should not learn anything from the data owned by the institution, so in particular, it should not learn which data the institution already had.

To realize this, a possibility is to use an unbalanced private set union protocol (UPSU), where a receiver inputs a large data set and a sender inputs a small data set. The protocol should output to the receiver the union of the two sets, without learning anything on the intersection of the sets, while the sender should not learn anything.

Our main goals in the conception of an UPSU are to achieve a communication volume that is linear in the size of the small set, and a computation cost for the sender independent of the size of the receiver's set. We reach those goals using efficient homomorphic algorithm on polynomials. Up to our knowledge, we present the first UPSU in which the communication volume is independent of the size of the receiver's set.

1 Introduction

Suppose a whistleblower knows some confidential and compromising data on some entity and would like to share those to a supranational institution or a newspaper. First of all, if the whistleblower sends those data, the message has to be encrypted, since an adversary should not be able to intercept the conversation and learn any information. Also, if the institution or the newspaper receives the entire whistleblower's data set, maybe it already possesses parts of this in its own data base, and this could leak some information about the whistleblower (and therefore compromise its anonymity). We thus focus on protocols computing the union of sets, that only gives to the receiver the sender's data that was not already known.

*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

More formally, the receiver should learn the union of the data set, without learning anything on the set intersection. Another naturally expected security property is that the whistleblower should not learn anything from the data set owned by the institution or the newspaper. Such a protocol is called a private set union (PSU). It 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 desired functionality of such a protocol 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 with (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 $\mathbf{X}$ and the receiver $\mathcal{R}$ learns nothing about the contents of $\mathbf{X} \cap \mathbf{Y}$. Recently, several private set union protocols have been

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

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

designed [Brickell and Shmatikov(2005), Kissner and Song(2005), Frikken(2007), Davidson and Cid(2017), Kolesnikov et al.(2019), Garimella et al.(2021a), Jia et al.(2022), Zhang et al.(2023)], motivated by numerous other practical applications such as IP blacklist and vulnerability data aggregation, or disease data collection on hospitals.

In our application, the data set owned by the whistleblower as well as the computing power may be (vastly) smaller than the receiver's. We are therefore interested in the case of *unbalanced* private set union (UPSU), and the goals of such a protocol are to reduce as much as possible the communication size and the computational cost of the sender. Note that it is impossible to reduce the communication size below that of the sender's set, since in the worst case its entire set must be revealed to the receiver, while the sender must not know how many elements were actually revealed.

More precisely, the focus of this work is the following: we define m and n to be the respective set sizes of the sender $\mathcal{S}$ and the receiver $\mathcal{R}$, we will assume $n > m$ 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). We present three new UPSU protocols and compare our asymptotics with the ones of three PSU protocol adapted in the unbalanced setting and the UPSU protocol from [Tu et al.(2023)] which is the only one that recently dealt with the unbalanced case. In [Frikken(2007)], the receiver represents its set $\mathbf{X} = \{x_j\}_{j\in[n]}$ with a polynomial $P_{\mathcal{R}} = \prod_{j=1}^{n}(Z - x_j) = a_0 + a_1 Z + \cdots + a_n Z^n$, whose roots are its elements, and sends $\{\widehat{a_j}\}_{j\in[m]}$, an encryption of this polynomial coefficients under a LHE scheme [Yi et al.(2014)], to the sender. The later evaluates homomorphically this encrypted polynomial in each elements of its set $\mathbf{Y} = \{y_i\}_{i\in[m]}$, obtaining the set $\{\widehat{P_{\mathcal{R}}(y_i)}\}_{i\in[m]}$, and sends to the receiver the encrypted tuples $\{\left(\widehat{P_{\mathcal{R}}(y_{\pi(i)})}, \widehat{P_{\mathcal{R}}(y_{\pi(i)})y_{\pi(i)}}\right)\}_{i\in[m]}$, computed homomorphically with π a random permutation. While decrypting those tuples, the receiver can compute only and exactly the $y \in \mathbf{Y} \setminus \mathbf{X}$. In [Davidson and Cid(2017)], the receiver represents its set $\mathbf{X}$ with a bloom filter [Bloom(1970)], which is a binary probabilistic data structure using hash functions. Given a bloom filter representing a set, one can check if an element is stored in it. Each hash of this element gives a position in the bloom filter. If there is a 0 in one of those positions in the bloom filter, then the element is not in the original set, and if all the positions lead to a 1, then the element is in the set, with overwhelmed probability. The receiver inverts each bloom filter entries (a 0 becomes a 1 and inversely) and encrypt those under a LHE scheme before sending the resulting $EIBF_{\mathbf{X}}$ to the sender. The sender obtains a set of positions by hashing each of its elements and adds homomorphically the entries of $EIBF_{\mathbf{X}}$ corresponding to those positions, obtaining an encrypted value $\widehat{v_y}$ for each $y \in \mathbf{Y}$. For each $y \in \mathbf{Y}$, the latter sends the encrypted tuple $(\widehat{v_y}, \widehat{v_y y})$, computed homomorphically, to the receiver whose can compute only and exactly the $y \in \mathbf{Y} \setminus \mathbf{X}$ after decryption. In [Zhang et al.(2023)], the PSU protocol is built from two subprotocols, namely a multi-query reverse private membership test (mq-RPMT) protocol, followed by an oblivious transfer (OT) protocol [Chou and Orlandi(2015)]. The first

one takes as inputs the set $\mathbf{X} = \{x_j\}_{j\in[n]}$ from the receiver, and the set $\mathbf{Y} = \{y_i\}_{i\in[m]}$ from the sender, and returns to the receiver a bit-vector $\mathbf{b} = \begin{pmatrix} b_1 & \cdots & b_m \end{pmatrix}$ such that $b_i = 1 \Leftrightarrow y_i \in \mathbf{X}$. It is build from an oblivious key-value stores (OKVS) [Garimella et al.(2021b)], which is a data structure allowing the access to an elements if and only if its associated key is known. With the oblivious transfer, the receiver obtains the $y_i \in \mathbf{Y}$ for which $b_i = 1$. In [Tu et al.(2023)], the global idea is to evaluate some polynomials representing the receiver's set, with roots in $\mathbf{X}$, in the sender's elements $y_i \in \mathbf{Y}$ as in [Frikken(2007)]. It is done using optimizations from a PSI protocol [Chen et al.(2018)] to reduce the multiplicative depth and the efficiency. Namely, using hash tables, windowing, batching, oblivious transfer and fully homomorphic encryption scheme. It introduces a new cryptographic protocol called permuted matrix private equality test (pm-PEQT) that compares two matrices $R = (r_{ij})_{ij}$ and $R' = (r'_{ij})_{ij}$, and gives to the receiver a binary matrix $B = (b_{ij})_{ij}$ such that $b_{ij} = 1 \Leftrightarrow r_{\pi(ij)} = r'_{\pi(ij)}$, for π a permutation on columns and row chosen by the sender. The pm-PEQT is used to compare a matrix $(P_{ij}(y_j) + r_{ij})_{ij}$, containing polynomial evaluations masked with a random value, to the matrix $(r_{ij})_{ij}$ containing only the random values. When one entry of the resulting binary matrix is 1, $b_{ij} = 1$, it means that $y_{\pi(j)}$ is a root of $P_{\pi(ij)}$, so $y_{\pi(j)} \in \mathbf{X}$. The protocol ends with an oblivious transfer, giving to the receiver only the $y \in \mathbf{Y}$ that are not root of any P_{ij}. However, as mentioned in [Kolesnikov et al.(2019)], the usage of hash tables leaks some clue on the intersection set $\mathbf{X} \cap \mathbf{Y}$.

In Table 1, the "Comm. Vol." column represents a bound on the quantity of exchanged elements, and the "Cost" columns represent a bound on the number of arithmetic operations performed by each party. For all the protocols using fully homomorphic encryptions, we also show the multiplicative depth of the protocols as it has a huge impact on the performances in practice. A value colored in green is a value satisfying our goals (which are a sender cost, and a communication volume, independent of the size of the receiver's larger set). Orange and red values are used to denote larger complexity bounds, that is logarithmic or worse, respectively in the size of the larger set.

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

Protocol	Cost for $\mathcal{R}$	Cost for $\mathcal{S}$	Comm. Vol.	# rounds	Depth	Security
[Frikken(2007)]	$O(n^{1+\epsilon})$	$O(nm)$	$O(n)$	2		✓
[Davidson and Cid(2017)]	$O(n)$	$O(m \log n)$	$O(n)$	2		✓
[Zhang et al.(2023)]	$O(n)$	$O(m \log n)$	$O(n)$	2+OT		✓
[Tu et al.(2023)]	$O(n)$	$O(m \log n)$	$O(m \log n)$	$\geq$ 4+OT	$\leq \log\log(n/m)$	✗
Protocol 1	$O(mn)$	$O(m)$	$O(m)$	3	$\log n + 1$	✓
Protocol 2	$O(n^{1+\epsilon})$	$O(m^{1+\epsilon})$	$O(m)$	3	$2(\log n - \log m) + 1$	✓
Protocol 3	$O(n^{1+\epsilon})$	$O(m^{1+\epsilon})$	$O(m)$	3	$2\log n + 1$	✓

Our contributions. We present three new (related) protocols for unbalanced PSU that all have a communication volume linear in the size of the sender's set, which is a first, up to our knowledge. Both protocols combine two different encryption schemes, namely a linearly and a fully homomorphic encryption schemes, and introduce efficient homomorphic algorithm on polynomials, which can be of independent interest. The main idea is always to evaluate homomorphically a polynomial, whose roots are the receiver's elements, in each of the sender's elements. What differentiate our protocols are the following aspects.

- The first protocol relies on the practical aspect of an instantiation of a FHE scheme, namely the HElib[1] instantiation of the BGV cryptosystem. This protocol is optimal in terms of communication volume and sender's arithmetic cost, keeps the multiplicative depth low, and we propose an implementation of it, together with our practical results. It is built from a fully homomorphic multi-point evaluation, using homomorphic scalar products, batching and parallelism.
- The second one is a theoretical protocol, that still has an optimal communication volume, but asymptotically reduces the arithmetic cost of the receiver, while slightly increasing the sender's. It also

[1] https://github.com/homenc/HElib

reduces the multiplicative depth when $\sqrt{n} < m < n$. It relies on efficient fully homomorphic euclidean remainder and linearly homomorphic multi-point evaluation algorithms, which are of independent interest.

- The last protocol is theoretically optimal in communication volume and arithmetic cost for the sender, has a low asymptotic arithmetic cost for the receiver, allows compatibility between LHE and FHE, but increases slightly the multiplicative depth. It relies on an efficient fully homomorphic multi-point evaluation algorithm, that is of independent interest.

Outline. In Section 2, we introduce the adversary model and the security assumptions, and we propose a formal definition of an unbalanced private set union protocol. The Section 3 defines the homomorphic encryption schemes, namely the linearly and the fully, together with the notation used in this paper. In Section 4, we present our optimal communication and implemented protocol, using homomorphic batched scalar multi-point evaluation, and we expose our practical results on computational timings and communication volume. Finally, in Section 5 we present two theoretical protocol variants, using efficient homomorphic algorithm on polynomials, that improve some aspects of the asymptotic.

2 Security Model and UPSU definition

2.1 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

$$\mathbf{view}_i^{\Pi}(x,y) := (w, C_i, M_i) \tag{1}$$

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

$$\mathbf{output}^{\Pi}(x,y) = (\mathbf{output}_1^{\Pi}(x,y), \mathbf{output}_2^{\Pi}(x,y)) \tag{2}$$

Definition 2.1. 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 $I, J \subset \{0,1\}^*$:

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

where $\overset{c}{\equiv}$ denotes the computational indistinguishability [Goldreich(2001)].

2.2 Unbalanced Private Set Union Scheme

In this section, we want to propose a formal definition of an unbalanced private set union (UPSU) protocol divided in five algorithms. To begin, there are two main constructions of (not unbalanced) private set union

protocol. The first one uses public key homomorphic encryption (HE). The receiver got the secret key of an HE, encrypts its set $\mathbf{X}$, or a representation of it (for example as a polynomial in [Frikken(2007)] or a bloom filter in [Davidson and Cid(2017)]) and sends it to the sender. The later performs an homomorphic evaluation on the received encryption in each of its set elements $y \in \mathbf{Y}$. This leads to a set of ciphertexts $\{\widehat{e_i}\}_i$ such that $\widehat{e_i}$ is an encryption of 0 if and only if the evaluation point y_i is in $\mathbf{X}$. Finally, the sender sends the encrypted tuples $\{(\widehat{e_i}, \widehat{e_i y_i})\}_i$ to the receiver, that can compute, after decryption, only the $y_i \in \mathbf{Y} \setminus \mathbf{X}$, and obtain the union. The second construction is basically based on two sub-protocols using symmetric-key operations. The first one attempts to give, with inputs the set of each party, to the receiver a bit-vector (for example the pm-PEQT protocol from [Zhang et al.(2023)]) such that an entry 1 in this vector at position i means that the i^{th} sender's element is not in the receiver's set. Given this bit-vector, the receiver asks obliviously all the y_i such that the vector has a 1 in position i, using an oblivious transfer protocol. In both cases, the receiver sends the first message in the protocol, which implies that the first message as a size proportional to the receiver's set size, and obviously, it also has to receive the last message. The first construction is usually done in two rounds, without taking in account the setup of the protocol, while the second needs some extra rounds for the oblivious transfer. In both cases, a PSU protocol can be divided into four steps: A setup phase, in which there is a hand-check on the context (on the encryption scheme, on the hash functions used , ...) and the potential keys are exchanged; An encoding phase, in which the receiver sends a representation, an encoding, of its set; An evaluation phase, in which the sender will evaluate the message received in its elements' set, leading to a data set given to the receiver. This data set is either a set $\{(\widehat{e_i}, \widehat{e_i y_i})\}_i$, or a bit-vector, depending on the construction; And finally, an union phase, allowing the receiver to obtain the union set, either directly in the first construction, of after an oblivious transfer. For an UPSU protocol, one goal is to keep the communication volume low, so the small set owner, the sender, has to send the first message. As the receiver has to obtain the last message, it implies that an UPSU has an odd number of rounds. It is not possible to have a honest-but-curious PSU protocol in one single round, because the receiver could obtain the full sender's set; indeed, the receiver can use the message received an perform the protocol with the empty set as input. This leads to the fact that we need at least 3 rounds, and one more step than in a PSU protocol. We divide an UPSU protocol in five algorithms, namely **Setup**, **Encode**, **Reduce**, **Map** and **Union**. For a sender $\mathcal{S}$ that owns a set $\mathbf{Y} \subset \mathbb{M}$ and a receiver $\mathcal{R}$ that owns a set $\mathbf{X} \subset \mathbb{M}$:

- $\{keys_{\mathcal{R}}, keys_{\mathcal{S}}\} \leftarrow \mathbf{Setup}(\kappa, \lambda)$: On input of a computational security parameter κ and optionally a statistical security parameter λ, set up a context (encryption schemes, hash functions, ...) with respect to κ, λ, and outputs keys $keys_{\mathcal{R}}$ to the receiver and keys $keys_{\mathcal{S}}$ to the sender;
- $E_{\mathbf{Y}} \leftarrow \mathbf{Encode}(\mathbf{Y}, keys_{\mathcal{S}})$: Given sender's set $\mathbf{Y}$ and keys $keys_{\mathcal{S}}$, outputs $E_{\mathbf{Y}}$ to the receiver, an encoding of the set $\mathbf{Y}$;
- $R_{\mathbf{X}|E_{\mathbf{Y}}} \leftarrow \mathbf{Reduce}(\mathbf{X}, E_{\mathbf{Y}}, keys_{\mathcal{R}})$: As input, takes receiver's set $\mathbf{X}$, keys $keys_{\mathcal{R}}$ and $E_{\mathbf{Y}}$, an encoding of the set $\mathbf{Y}$. Outputs $R_{\mathbf{X}|E_{\mathbf{Y}}}$ to the sender, an encoding of the set $\mathbf{X}$, reduced in size depending on $E_{\mathbf{Y}}$;
- $M_{Y|R_{\mathbf{X}}} \leftarrow \mathbf{Map}(\mathbf{Y}, R_{\mathbf{X}|E_{\mathbf{Y}}}, keys_{\mathcal{S}})$: On input of a set $\mathbf{Y}$, an encoding $R_{\mathbf{X}|E_{\mathbf{Y}}}$ and a set of keys $keys_{\mathcal{S}}$, outputs to the receiver an encoded data set $M_{Y|R_{\mathbf{X}}}$ representing the set $\mathbf{Y} \setminus \mathbf{X}$, depending on $\mathbf{Y}$ and $R_{\mathbf{X}|E_{\mathbf{Y}}}$;
- $\mathbf{Z} \leftarrow \mathbf{Union}(\mathbf{X}, M_{Y|R_{\mathbf{X}}}, keys_{\mathcal{R}})$: On input of the receiver's set $\mathbf{X}$, an encoded data set $\mathcal{D}$ representing the set $\mathbf{Y} \setminus \mathbf{X}$ and a set of keys $keys_{\mathcal{R}}$, outputs the union set $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$ to the receiver.

Definition 2.2. (**Setup**, **Encode**, **Reduce**, **Map**, **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

$$\begin{aligned} \{keys_{\mathcal{R}}, keys_{\mathcal{S}}\} &\leftarrow \mathbf{Setup}(\kappa, \lambda) \\ E_{\mathbf{Y}} &\leftarrow \mathbf{Encode}(\mathbf{Y}, keys_{\mathcal{S}}) \\ R_{\mathbf{X}|E_{\mathbf{Y}}} &\leftarrow \mathbf{Reduce}(\mathbf{X}, E_{\mathbf{Y}}, keys_{\mathcal{R}}) \end{aligned}$$

then the scheme is correct if

$$\mathbf{Union}(\mathbf{X}, \mathbf{Map}(\mathbf{Y}, R_{\mathbf{X}|E_{\mathbf{Y}}}, keys_{\mathcal{S}}), keys_{\mathcal{R}}) = \mathbf{X} \cup \mathbf{Y}. \tag{3}$$

ii) **Privacy.** The scheme ensures privacy of each participant's set if it is secured following Def. 2.1 where the definition is instantiated with the PPT functionality

$$f : \mathcal{P}(\mathbb{M}) \times \mathcal{P}(\mathbb{M}) \longrightarrow (\mathcal{P}(\mathbb{M}) \times \mathbb{N}) \times \mathcal{P}(\mathbb{M}) \tag{4}$$

defined by

$$f(\mathbf{X}, \mathbf{Y}) = ((\mathbf{X} \cup \mathbf{Y}, |\mathbf{Y}|), \emptyset). \tag{5}$$

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 $o(|\mathbf{X}|)$.

Remark 1. Note that a non-unbalanced PSU protocol can be described with those algorithms, considering that **Encode** outputs $\emptyset$. However, generally such a protocol will not satisfy the **Unbalanced efficiency** of the definition. Up to our knowledge, only [Tu et al.(2023)] is made for the unbalanced situation, and it fits this definition.

3 Cryptographic tools: Homomorphic encryption schemes

We first introduce the main cryptographic tools used in our protocol, namely Linearly and Fully Homomorphic Encryption Schemes.

3.1 Linearly Homomorphic Encryption Scheme

Notation: For sake of clarity, in the following $\widehat{x}$ denotes a value encrypted using linearly homomorphic encryption (LHE).
For our purposes, an 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 public and secret keys (pk, sk). pk implicitly defines a finite ring[2] $\mathbb{M}_L$, which is the plaintext space, and a ciphertext space $\mathbb{E}_L$;
- $\widehat{m} \leftarrow \mathbf{L.E}_{pk}(m)$: Given as inputs a plaintext $m \in \mathbb{M}_L$ and a public key pk, outputs a ciphertext $\widehat{m} \in \mathbb{E}_L$;
- $m \leftarrow \mathbf{L.D}_{sk}(\widehat{m})$: Given as inputs a ciphertext $\widehat{m} \in \mathbb{E}_L$ and a secret key sk, outputs a plaintext $m \in \mathbb{M}_L$;
- $\widehat{m_3} \leftarrow \widehat{m_1} +_L \widehat{m_2}$: Given as inputs two ciphertexts $\widehat{m_1}, \widehat{m_2} \in \mathbb{E}_L$, outputs a ciphertext $\widehat{m_3} \in \mathbb{E}_L$;

[2]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.

- $\widehat{m_3} \leftarrow m_1 \ltimes_L \widehat{m_2}$: Given as inputs a plaintext $m_1 \in \mathbb{M}_L$ and a ciphertext $\widehat{m_2} \in \mathbb{E}_L$, outputs a ciphertext $\widehat{m_3} \in \mathbb{E}_L$.

Definition 3.1. $(\mathbf{L.Setup}, \mathbf{L.E}, \mathbf{L.D}, +_L, \ltimes_L)$ 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 $m, m_1, m_2 \in \mathbb{M}_L$,

$$\mathbf{L.D}_{sk}(\mathbf{L.E}_{pk}(m)) = m, \tag{6}$$
$$\mathbf{L.D}_{sk}(\mathbf{L.E}_{pk}(m_1) +_L \mathbf{L.E}_{pk}(m_2)) = m_1 + m_2, \tag{7}$$
$$\mathbf{L.D}_{sk}(m_1 \ltimes_L \mathbf{L.E}_{pk}(m_2)) = m_1 m_2. \tag{8}$$

ii) **Security.** The scheme is IND-CPA secure.

Remark 2. We extend naturally the encryption and decryption algorithms for a LHE to allow vectors as inputs: if $\mathbf{v} = \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix} \in {\mathbb{M}_L}^n$, $\mathbf{L.E}_{pk}(\mathbf{v})$ outputs $\widehat{\mathbf{v}} = \begin{pmatrix} \widehat{v_1} & \cdots & \widehat{v_n} \end{pmatrix} \in {\mathbb{E}_L}^n$ such that $\widehat{v_i}$ is an encryption of v_i for $1 \leq i \leq n$. Similarly, $\mathbf{L.D}_{sk}(\widehat{\mathbf{v}})$ outputs $\mathbf{v}$. In the same way, we extend these algorithms to polynomial inputs and outputs in $\mathbb{M}_L[Z]$ or $\mathbb{E}_L[Z]$ by stating that the encryption of a polynomial is the encryption of its vector of coefficients. This allows for instance to extend the algorithm $+_L$ to vectors or polynomials. Also, we can extend $\ltimes_L$ to a matrix-vector or a vector-matrix product, where the left part is in clear and the right part is encrypted:

$$\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & & \vdots \\ v_{m1} & \cdots & v_{mn} \end{pmatrix} \ltimes_L \begin{pmatrix} \widehat{c_1} \\ \vdots \\ \widehat{c_n} \end{pmatrix} = \begin{pmatrix} v_{11} \ltimes_L \widehat{c_1} +_L \cdots +_L v_{1n} \ltimes_L \widehat{c_n} \\ \vdots \\ v_{m1} \ltimes_L \widehat{c_1} +_L \cdots +_L v_{mn} \ltimes_L \widehat{c_n} \end{pmatrix} \tag{9}$$

$$\begin{pmatrix} v_1 & \cdots & v_m \end{pmatrix} \ltimes_L \begin{pmatrix} \widehat{c_{11}} & \cdots & \widehat{c_{1n}} \\ \vdots & & \vdots \\ \widehat{c_{m1}} & \cdots & \widehat{c_{mn}} \end{pmatrix} = \begin{pmatrix} v_1 \ltimes_L \widehat{c_{11}} +_L \cdots +_L v_m \ltimes_L \widehat{c_{m1}} \\ \vdots \\ v_1 \ltimes_L \widehat{c_{1n}} +_L \cdots +_L v_m \ltimes_L \widehat{c_{mn}} \end{pmatrix}^t \tag{10}$$

In particular, eq. (9) and (10) imply that we can compute the homomorphic polynomial product between a plaintext polynomial A and a ciphertext polynomial $\widehat{C}$. This corresponds to the following linear maps:

$$\widehat{C} \longmapsto A \ltimes_L \widehat{C} \tag{11}$$
$$A \longmapsto A \ltimes_L \widehat{C} \tag{12}$$

3.2 Fully Homomorphic Encryption Scheme

Notation: For sake of clarity, in the following $\widetilde{x}$ denotes a value encrypted using fully homomorphic encryption (FHE).
For our purposes, an FHE scheme consists of six algorithms

$$(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F) \tag{13}$$

where $(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F)$ is a LHE, where the plaintext space is a finite ring $\mathbb{M}_F$[3] and the ciphertext space is $\mathbb{E}_F$, and the algorithm $\times_F$ is as follows:

[3]More generally FHE may be defined over infinite rings, depending on the scheme used. For our purposes, we need exact polynomial arithmetic on finite rings.

- $\widetilde{m_3} \leftarrow \widetilde{m_1} \times_F \widetilde{m_2}$: Given as inputs two ciphertexts $\widetilde{m_1}$, $\widetilde{m_2} \in \mathbb{E}_F$, outputs a ciphertext $\widetilde{m_3} \in \mathbb{E}_F$.

Definition 3.2. $(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F)$ is a semantically secure FHE if it satisfies the following properties:

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

$$\mathbf{F.D}_{sk}(\mathbf{F.E}_{pk}(m_1) \times_F \mathbf{F.E}_{pk}(m_2)) = m_1 m_2. \tag{14}$$

ii) **Security.** The scheme is IND-CPA secure.

Remark 3. Similarly to the LHE scheme, we extend $(\mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F)$ to vector and polynomial operations.

4 Optimal communication volume, low depth, batchable and parallelizable UPSU protocol

In practice, using FHE operations imply an important computational cost so it requires to use some practical tools to make those operations doable in real life. On one hand, the actual FHE schemes are built with leveled homomorphic encryption schemes, that allow one to perform arithmetic circuits up to a certain depth, that we will denote $d_{\max}$. In fact, a homomorphic product on a ciphertext is done together with a *modulus switching* procedure, which reduces its size, up to the minimum size after $d_{\max}$ such procedure. Then, such a schemes is upgraded into a fully homomorphic scheme, allowing unbounded depth circuits, with the help of a so-called *boostrapping* procedure. The latter is rarely implemented in the homomorphic libraries and is extremely costly in time computation. That explains why we will always try to control the multiplicative depth of our protocols to keep it, in practice, below $d_{\max}$. The maximum depth allowed can be controlled while choosing a FHE context, but there is a trade-off: to increase $d_{\max}$ it is needed to increase the size of the ciphertexts, to allow more modulus switching, and it leads to a lower computational security of the encryption. On the other hand, it is possible to batch together a certain amount m of ciphertexts and a homomorphic operation on the resulting batched ciphertext acts slot-wise. The choice of the context, and in particular of the plaintext space, allow to choose the size m of a batch; basically, it is a Chinese remainder. In the same vein, it is a good idea to use as much as possible the parallelism in order to reduce the computational cost. For those reasons, our first UPSU protocol attempts to have an optimal communication volume, while keeping the multiplicative depth of the fully homomorphic operations low and using as well as possible the batching. This protocol is also highly parallelizable; the time computation without parallelization is almost divided by the number of threads. The idea is to evaluate homomorphically the polynomial $P_{\mathcal{R}}$, whose roots are the receiver's elements, in all the sender's elements using homomorphic scalar product. To keep the security and to reduce the computational cost of the sender[4], we make a transition between an FHE scheme and an LHE scheme.

4.1 Practical tools in FHE

For our purpose and for the clarity of our protocol, we define two utility algorithms. The batching algorithm **F.Batch** allows to batch together m ciphertexts, and the algorithm **low** performs modulus switchings on a ciphertext in order to reduce as much as possible its size. Note that those algorithms are intrinsic to the usage of FHE, so in order to stay coherent with our asymptotic comparisons with schemes that does not

[4]LHE computations are faster than FHE computations.

use it, we consider the following. Performing a homomorphic operation on a batch ciphertext is equivalent to perform m homomorphic operations, the volume of a batched ciphertext is the volume of m ciphertexts, and, as the communication volume considered in the protocols counts the number of ciphertext exchanged, the usage of **low** does not change the asymptotic.

More formally, we define the batching as the following algorithm.

Definition 4.1. Let $(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F)$ be a FHE scheme allowing the batch of m ciphertexts. The fully homomorphic batching is an algorithm **F.Batch** such that

- $\widetilde{\mathbf{y}} \leftarrow \mathbf{F.Batch}(\begin{pmatrix}\widetilde{y_1} & \cdots & \widetilde{y_m}\end{pmatrix})$: Given as input a vector of m ciphertexts $\begin{pmatrix}\widetilde{y_1} & \cdots & \widetilde{y_m}\end{pmatrix} \in {\mathbb{E}_F}^m$, outputs a batched ciphertext $\widetilde{\mathbf{y}} \in {\mathbb{E}_F}^m$.

We extend the definitions of $(\mathbf{F.D}, +_F, \ltimes_F, \times_F)$ to the batched ciphertexts.
Correctness. For a security parameter κ, if $(pk, sk) \leftarrow \mathbf{F.Setup}(\kappa)$, for $\begin{pmatrix}y_1 & \cdots & y_m\end{pmatrix}, \begin{pmatrix}z_1 & \cdots & z_m\end{pmatrix} \in {\mathbb{M}_F}^m$, let

$$\widetilde{\mathbf{y}} \leftarrow \mathbf{F.Batch}\left(\begin{pmatrix}\mathbf{F.E}_{pk}(y_1) & \cdots & \mathbf{F.E}_{pk}(y_m)\end{pmatrix}\right), \tag{15}$$

$$\widetilde{\mathbf{z}} \leftarrow \mathbf{F.Batch}\left(\begin{pmatrix}\mathbf{F.E}_{pk}(z_1) & \cdots & \mathbf{F.E}_{pk}(z_m)\end{pmatrix}\right). \tag{16}$$

Then, for a plaintext $k \in \mathbb{M}_F$,

$$\mathbf{F.D}_{sk}(\widetilde{\mathbf{y}} +_F \widetilde{\mathbf{z}}) = \begin{pmatrix}y_1 + z_1 & \cdots & y_m + z_m\end{pmatrix}, \tag{17}$$

$$\mathbf{F.D}_{sk}(\widetilde{\mathbf{y}} \times_F \widetilde{\mathbf{z}}) = \begin{pmatrix}y_1 z_1 & \cdots & y_m z_m\end{pmatrix}, \tag{18}$$

$$\mathbf{F.D}_{sk}(k \ltimes_F \widetilde{\mathbf{y}}) = \begin{pmatrix}k y_1 & \cdots & k y_m\end{pmatrix}. \tag{19}$$

We also define the algorithm **low** that reduces the size of a ciphertext by performing modulus switching.

Definition 4.2. Let $(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F)$ be a FHE scheme allowing a circuit of multiplicative depth up to $d_{\max}$ before needing the boostrapping procedure. The algorithm **low** is defined as follows.

- $\widetilde{z} \leftarrow \mathbf{low}(\mathbf{y})$: Given as input a ciphertext $\widetilde{y} \in \mathbb{E}_F$, outputs a batched ciphertext $\widetilde{z} \in \mathbb{E}_F$.

Correctness. For a security parameter κ, if $(pk, sk) \leftarrow \mathbf{F.Setup}(\kappa)$, for a batched ciphertext $\widetilde{y} \in \mathbb{E}_F$, if

$$\widetilde{z} \leftarrow \mathbf{low}(\widetilde{y}), \tag{20}$$

then,

$$\mathbf{F.D}_{sk}(\widetilde{y}) = \mathbf{F.D}_{sk}(\widetilde{z}) \tag{21}$$

and $\widetilde{z}$ is at depth $d_{\max}$[5].

Remark 4. We extend naturally the algorithm **low** to allow ciphertext polynomials, ciphertext vectors and batched vectors.

4.2 Fully homomorphic batched scalar multi-point evaluation

We will divide the homomorphic multi-point evaluation in three subalgorithms: the first one generates homomorphically a vector of the powers of the batched evaluation points, the second one evaluates multiple polynomials with a homomorphic scalar product, and the third one multiplies homomorphically the evaluation values obtained previously.

[5]Doing one more homomorphic product of this ciphertext, without doing a boostrap, will make it not longer decipherable.

Definition 4.3. Let $(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F)$ be a FHE scheme. We introduce the algorithms $\mathbf{F.Pow}, \mathbf{F.Scal}, \mathbf{F.Prod}$ such that

- $\overrightarrow{\widetilde{\mathbf{v}}} \leftarrow \mathbf{F.Pow}(\widetilde{\mathbf{y}}, t)$: Given as input a batched ciphertext $\widetilde{\mathbf{y}} \in {\mathbb{E}_F}^m$ and a positive integer t, outputs a vector of $t+1$ batched ciphertexts $\overrightarrow{\widetilde{\mathbf{v}}} \in ({\mathbb{E}_F}^m)^{t+1}$.
- $\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\} \leftarrow \mathbf{F.Scal}\left(\{P_1, ..., P_k\}, \overrightarrow{\widetilde{\mathbf{v}}}\right)$: Given as inputs a set of k plaintext polynomials of degrees t, $\{P_1, ..., P_k\} \subset \mathbb{M}_F[Z]$, and a vector of $t+1$ batched ciphertexts $\overrightarrow{\widetilde{\mathbf{v}}} \in ({\mathbb{E}_F}^m)^{t+1}$, outputs a set of k batched ciphertexts $\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\} \subset {\mathbb{E}_F}^m$.
- $\widetilde{\mathbf{e}} \leftarrow \mathbf{F.Prod}\left(\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\}\right)$: Given as input a set of k batched ciphertexts $\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\} \subset {\mathbb{E}_F}^m$, outputs a batched ciphertext $\widetilde{\mathbf{e}} \in {\mathbb{E}_F}^m$.

Those algorithms satisfy the following correctness properties.
Correctness. For a security parameter κ, let $(pk, sk) \leftarrow \mathbf{F.Setup}(\kappa)$. For $\begin{pmatrix} y_1 & \cdots & y_m \end{pmatrix} \in {\mathbb{M}_F}^m$ and a positive integer t, and for

$$\begin{pmatrix} \widetilde{\mathbf{v}_0} & \cdots & \widetilde{\mathbf{v}_t} \end{pmatrix} \leftarrow \mathbf{F.Pow}(\mathbf{F.Batch}\left(\begin{pmatrix} \mathbf{F.E}_{pk}(y_1) & \cdots & \mathbf{F.E}_{pk}(y_m) \end{pmatrix}\right), t), \tag{22}$$

then, for all $i \in \{0, ..., t\}$,

$$\mathbf{F.D}_{sk}(\widetilde{\mathbf{v}_i}) = \begin{pmatrix} y_1^i & \cdots & y_m^i \end{pmatrix}. \tag{23}$$

For $\{P_1, ..., P_k\} \subset \mathbb{M}_F[Z]$, k polynomials of degrees t, and a vector of $t+1$ batched ciphertexts $\overrightarrow{\widetilde{\mathbf{v}}} := \begin{pmatrix} \widetilde{\mathbf{v}_0} & \cdots & \widetilde{\mathbf{v}_t} \end{pmatrix} \in ({\mathbb{E}_F}^m)^{t+1}$. For $i \in \{0, ..., t\}$, if

$$\mathbf{F.D}_{sk}(\widetilde{\mathbf{v}_i}) = \begin{pmatrix} y_{1,i} & \cdots & y_{m,i} \end{pmatrix} \tag{24}$$

and

$$\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\} \leftarrow \mathbf{F.Scal}\left(\{P_1, ..., P_k\}, \overrightarrow{\widetilde{\mathbf{v}}}\right), \tag{25}$$

then, for $j \in \{1, ..., k\}$, denoting $P_j = \sum_{i=0}^{t} p_{j,i} Z^i$, we have

$$\mathbf{F.D}_{sk}(\widetilde{\mathbf{e}_j}) = \begin{pmatrix} \sum_{i=0}^{t} p_{j,i} y_{1,i} & \cdots & \sum_{i=0}^{t} p_{j,i} y_{m,i} \end{pmatrix}. \tag{26}$$

For k batched ciphertexts $\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\} \subset {\mathbb{E}_F}^m$ such that, for $j \in \{1, ..., k\}$,

$$\mathbf{F.D}_{sk}(\widetilde{\mathbf{e}_j}) = \begin{pmatrix} z_{1,j} & \cdots & z_{m,j} \end{pmatrix}, \tag{27}$$

then,

$$\mathbf{F.D}_{sk}\left(\mathbf{F.Prod}\left(\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\}\right)\right) = \begin{pmatrix} \prod_{j=1}^{k} z_{1,j} & \cdots & \prod_{j=1}^{k} z_{m,j} \end{pmatrix}. \tag{28}$$

Remark 5. In particular, let $P = P_1 ... P_k$ be a plaintext polynomial, product of k polynomials of degrees t in $\mathbb{M}_F[Z]$, and let $\{y_1, ..., y_m\} \subset \mathbb{M}$ be m plaintexts. Then, for a security parameter κ and $(pk, sk) \leftarrow \mathbf{F.Setup}(\kappa)$, for $\widetilde{\mathbf{y}} \leftarrow \mathbf{F.Batch}\left(\begin{pmatrix} \mathbf{F.E}_{pk}(y_1) & \cdots & \mathbf{F.E}_{pk}(y_m) \end{pmatrix}\right)$, we have the following property:

$$\mathbf{F.D}_{sk}\left(\mathbf{F.Prod}\left(\mathbf{F.Scal}\left(\{P_1, ..., P_k\}, \mathbf{F.Pow}\left(\widetilde{\mathbf{y}}, t\right)\right)\right)\right) = \begin{pmatrix} P(y_1) & \cdots & P(y_m) \end{pmatrix}. \tag{29}$$

According to the remark, we define the homomorphic batched scalar multi-point evaluation.

Definition 4.4. The homomorphic batched scalar multi-point evaluation **F.BSMEv** is an algorithm combining **F.Pow**, **F.Scal** and **F.Prod** as following. For a batched ciphertext $\widetilde{\mathbf{y}} \in {\mathbb{E}_F}^m$, $P = P_1...P_k$ a plaintext polynomial, product of k polynomials of degrees t in $\mathbb{M}_F[Z]$,

$$\mathbf{F.BSMEv}(P, \widetilde{\mathbf{y}}, t) = \mathbf{F.Prod}\left(\mathbf{F.Scal}\left(\{P_1, ..., P_k\}, \mathbf{F.Pow}\left(\widetilde{\mathbf{y}}, t\right)\right)\right). \tag{30}$$

This algorithm satisfies the correctness of Rem. 5.

Proposition 1. *Let $P \in \mathbb{M}_F[Z]$ be of degree n, such that it is a product of $\sqrt{n}$ plaintext polynomials of degrees $\sqrt{n}$, and let $\widetilde{\boldsymbol{y}} \in {\mathbb{E}_F}^m$ be a batched ciphertext. $\boldsymbol{F.BSMEv}(P, \widetilde{\boldsymbol{y}}, \sqrt{n})$ can be computed in $O(mn)$ arithmetic operations and a depth $\lceil \log n \rceil + 1$.*

Proof. For a batched ciphertext $\widetilde{\mathbf{y}} \in {\mathbb{E}_F}^m$ and a positive integer t, computing $\mathbf{F.Pow}(\widetilde{\mathbf{y}}, t)$ requires t batched homomorphic multiplications, so in total mt homomorphic product, and has a depth $\log t$. For $\overrightarrow{\widetilde{\mathbf{v}}} \in ({\mathbb{E}_F}^m)^{t+1}$ and $\{P_1, ..., P_k\}$ a set of plaintext polynomials of degrees t, computing $\mathbf{F.Scal}(\{P_1, ..., P_k\}, \overrightarrow{\widetilde{\mathbf{v}}})$ requires to compute k batched scalar product between a plaintext vector and a batched ciphertext vector of size t, which needs k times t homomorphic products between a plaintext and a batched ciphertext, so in total it requires ktm plaintext/ciphertext products, with a depth 1. Finally, through a binary tree, computing $\mathbf{F.Prod}(\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\})$, for $\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\}$ a set of k batched ciphertexts, requires k homomorphic product between batched ciphertexts, so in total km homomorphic products, with a depth $\log k$. The Table 2 resumes the costs. As $n = kt$, the depth is $\lceil \log n \rceil + 1$, but to reduce the arithmetic cost, we reduce the number of homomorphic products $\times_F$ by taking $k = t = \sqrt{n}$. □

Table 2: Cost anaysis of **F.BSMEv**

	Depth	**nbr homo. prod.**
$\mathbf{F.Pow}(\widetilde{\mathbf{y}}, t)$	$\log t$	$mt \times$ " $\times_F$ "
$\mathbf{F.Scal}(\{P_1, ..., P_k\}, \overrightarrow{\widetilde{\mathbf{v}}})$	1	$ktm \times$ " $\ltimes_F$ "
$\mathbf{F.Prod}(\{\widetilde{\mathbf{e}_1}, ..., \widetilde{\mathbf{e}_k}\})$	$\log k$	$mk \times$ " $\times_F$ "
$\mathbf{F.BSMEv}(P, \widetilde{\mathbf{y}}, t)$	$\log kt + 1$	$ktm \times$ " $\ltimes_F$ " $+ (k+t) \times$ " $\times_F$ "

4.3 Optimal communication volume, low depth, batchable and parallelisable UPSU protocol

In our protocol, the sender $\mathcal{S}$ owns a set of m plaintext elements $\{y_i\}_{i \in [m]} \subset \mathbb{M}$, while the receiver $\mathcal{R}$ owns a set of n FHE plaintext elements $\{x_i\}_{i \in [n]} \subset \mathbb{M}$, and we admit that $n > m$. For sake of clarity, we suppose that the LHE scheme and the FHE scheme share the same plaintext space $\mathbb{M}$, which is a finite field; in our implementations, we are is that situation, but in Section 4.5, we propose some slight modifications to keep the correctness of our protocol under other assumptions.

Formally, our protocol is built with the algorithms **Setup**, **Encode**, **Reduce**, **Map** and **Union** respectively presented in Algs. 1 to 5. A more visual version is presented in Protocol 1.

Proposition 2. *The protocol built with the algorithms **Setup**, **Encode**, **Reduce**, **Map** and **Union**, respectively presented in Algs. 1 to 5, is correct.*

Proof. We assume that the LHE and the FHE schemes are correct as well as the algorithms **F.Batch**, **low** and **F.BSMEv**. Considering the notations of the different algorithms. In **Reduce**, $\widetilde{\mathbf{e}}$ is a batched encryption of $\begin{pmatrix} P_{\mathcal{R}}(y_1) & \cdots & P_{\mathcal{R}}(y_m) \end{pmatrix}$, so $\widetilde{\mathbf{h}}$ is a batched encryption of $\begin{pmatrix} P_{\mathcal{R}}(y_1) + k_1 & \cdots & P_{\mathcal{R}}(y_m) + k_m \end{pmatrix}$. Then, in **Map**, we have, for all $i \in [m]$, $\widehat{e_{\pi(i)}}$ is an encryption of $P_{\mathcal{R}}(y_i) + k_i - k_i = P_{\mathcal{R}}(y_i)$ and so $\widehat{\eta_{\pi(i)}}$ is

Algorithm 1 Setup(κ)

Input: A security parameter κ.
Output: A pair of LHE keys (pk_L, sk_L) and a FHE public key pk_F, both schemes with atleast κ-bit security.
Output: A pair of FHE keys (pk_F, sk_F) and a LHE public key pk_L, both schemes with atleast κ-bit security.
Remark: Both keys implicitly define the finite field plaintext space $\mathbb{M}$, but potentially two different ciphertext spaces, $\mathbb{E}_L$ and $\mathbb{E}_F$. The FHE scheme allows a multiplicative depth $d_{\max}$ before the need of a bootstrap.

1: $\mathcal{R}$: compute $(pk_L, sk_L) \leftarrow \mathbf{L.Setup}(\kappa)$ and send pk_L to $\mathcal{S}$;
2: $\mathcal{S}$: compute $(pk_F, sk_F) \leftarrow \mathbf{F.Setup}(\kappa)$ and send pk_F to $\mathcal{R}$;
3: $\mathcal{R}$: **return** $keys_{\mathcal{R}} \leftarrow \{(pk_L, sk_L), pk_F\}$;
4: $\mathcal{S}$: **return** $keys_{\mathcal{S}} \leftarrow \{(pk_F, sk_F), pk_L\}$;

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

Input: A set of plaintext $\mathbf{Y} = \{y_i\}_{i\in[m]} \subset \mathbb{M}$ and $keys_{\mathcal{S}} = \{(pk_F, sk_F), pk_L\}$.
Output: A batched ciphertext $\widetilde{\mathbf{y}} \in {\mathbb{E}_F}^m$ such that $\mathbf{F.D}_{sk_F}(\widetilde{\mathbf{y}}) = \begin{pmatrix} y_1 & \cdots & y_m \end{pmatrix}$.

1: $\mathcal{S}$: compute $\{\widetilde{y_i}\}_{i\in[m]} \leftarrow \{\mathbf{F.E}_{pk_F}(y_i)\}_{i\in[m]}$;
2: $\mathcal{S}$: compute $\widetilde{\mathbf{y}} \leftarrow \mathbf{F.Batch}(\begin{pmatrix} \widetilde{y_1} & \cdots & \widetilde{y_m} \end{pmatrix})$ and send $\widetilde{\mathbf{y}}$ to $\mathcal{R}$;
3: $\mathcal{R}$: **return** $\widetilde{\mathbf{y}}$;

Algorithm 3 Reduce($\mathbf{X}, \widetilde{\mathbf{y}}, keys_{\mathcal{R}}$)

Input: A set of plaintext $\mathbf{X} = \{x_j\}_{j\in[n]} \subset \mathbb{M}$, a batched ciphertext $\widetilde{\mathbf{y}} \in {\mathbb{E}_F}^m$ and $keys_{\mathcal{R}} = \{(pk_L, sk_L), pk_F\}$.
Output: A batched ciphertext $\widetilde{\mathbf{h}} \in {\mathbb{E}_F}^m$, at depth $d_{\max}$, and a set of ciphertexts $\{\widehat{k_i}\}_{i\in[m]} \subset \mathbb{E}_L$, such that for $i \in [m]$, $\mathbf{F.D}_{sk_F}(\widetilde{\mathbf{h}})[i] - \mathbf{L.D}_{sk_L}(\widehat{k_i}) = \prod_{j\in[n]} (\mathbf{F.D}_{sk_F}(\widetilde{\mathbf{y}})[i] - x_j)$.

1: $\mathcal{R}$: compute $P_{\mathcal{R}} \leftarrow \prod_{j\in[n]} (Z - x_j)$;
2: **for all** $i \in [m]$ **do**
3: $\quad$ $\mathcal{R}$: randomly select $k_i \xleftarrow{\$} \mathbb{M}$;
4: $\quad$ $\mathcal{R}$: compute $\widehat{k_i} \leftarrow \mathbf{L.E}_{pk_L}(k_i)$;
5: $\quad$ $\mathcal{R}$: compute $\widetilde{k_i} \leftarrow \mathbf{F.E}_{pk_F}(k_i)$;
6: **end for**
7: $\mathcal{R}$: compute $\widetilde{\mathbf{k}} \leftarrow \mathbf{F.Batch}(\begin{pmatrix} \widetilde{k_1} & \cdots & \widetilde{k_m} \end{pmatrix})$;
8: $\mathcal{R}$: compute $\widetilde{\mathbf{e}} \leftarrow \mathbf{F.BSMEv}(P_{\mathcal{R}}, \widetilde{\mathbf{y}}, \sqrt{n})$;
9: $\mathcal{R}$: compute $\widetilde{\mathbf{h}} \leftarrow \widetilde{\mathbf{k}} +_F \widetilde{\mathbf{e}}$;
10: $\mathcal{R}$: compute $\widetilde{\mathbf{h}} \leftarrow \mathbf{low}(\widetilde{\mathbf{h}})$;
11: $\mathcal{R}$: send $\widetilde{\mathbf{h}}$ and $\{\widehat{k_i}\}_{i\in[m]}$ to $\mathcal{S}$;
12: $\mathcal{S}$: **return** $\{\widetilde{\mathbf{h}}, \{\widehat{k_i}\}_{i\in[m]}\}$;

an encryption of $P_{\mathcal{R}}(y_i)y_i$. Finally, in **Union**, $(e_j, \eta_j) = (0, 0)$ if and only if $y_{\pi^{-1}(j)}$ is a root of $P_{\mathcal{R}}$, which is equivalent to say that $y_{\pi^{-1}(j)} \in \mathbf{X}$. When $y_{\pi^{-1}(j)}$ is not a root of $P_{\mathcal{R}}$, e_j is invertible as we are in a field, and $y_{\pi^{-1}(j)} = \eta_j e_j^{-1}$. It means that the y_j added to $\mathbf{X}$ in **Union** are only and exactly the y_j that were not in the $\mathcal{R}$'s input set. □

Algorithm 4 $\mathbf{Map}(\mathbf{Y}, \{\widetilde{\mathbf{h}}, \{\widehat{k_i}\}_{i\in[m]}\}, keys_{\mathcal{S}})$

Input: A set of m plaintexts $\mathbf{Y} = \{y_i\}_{i\in[m]} \subset \mathbb{M}$, a batched ciphertext $\widetilde{\mathbf{h}} \in {\mathbb{E}_F}^m$, a ciphertext set $\{\widehat{k_i}\}_{i\in[m]} \subset \mathbb{E}_L$, and $keys_{\mathcal{S}} = \{(pk_F, sk_F), pk_L\}$.
Output: A set of ciphertext tuples $\{(\widehat{e_j}, \widehat{\eta_j})\}_{j\in[m]} \subset {\mathbb{E}_L}^2$, such that for all $i \in [m]$, $\mathbf{L.D}_{sk_L}(\widehat{e_{\pi(i)}}) = \mathbf{F.D}_{sk_F}(\widetilde{\mathbf{h}})[i] - \mathbf{L.D}_{sk_L}(\widehat{k_i})$ and $\mathbf{L.D}_{sk_L}(\widehat{\eta_{\pi(i)}}) = y_i \mathbf{L.D}_{sk_L}(\widehat{e_{\pi(i)}})$.

$\mathcal{S}$: compute $\begin{pmatrix} h_1 & \cdots & h_m \end{pmatrix} \leftarrow \mathbf{F.D}_{sk_F}(\widetilde{\mathbf{h}})$;
$\mathcal{S}$: randomly select $\pi \xleftarrow{\$} \mathfrak{S}_m$;
for all $i \in [m]$ **do**
    $\mathcal{S}$: compute $\widehat{e_{\pi(i)}} \leftarrow \mathbf{L.E}_{pk_L}(h_i) -_L \widehat{k_i}$;
    $\mathcal{S}$: compute $\widehat{\eta_{\pi(i)}} \leftarrow y_i \ltimes_L \widehat{e_{\pi(i)}}$;
end for
$\mathcal{S}$: send $\{(\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}})\}_{i\in[m]}$ to $\mathcal{R}$;
$\mathcal{R}$: **return** $\{(\widehat{e_j}, \widehat{\eta_j})\}_{j\in[m]}$;

Algorithm 5 $\mathbf{Union}(\mathbf{X}, \{(\widehat{e_j}, \widehat{\eta_j})\}_{j\in[m]}, keys_{\mathcal{R}})$

Input: A set of plaintexts $\mathbf{X} = \{x_j\}_{j\in[n]} \subset \mathbb{M}$, a set of ciphertext tuples $\{(\widehat{e_j}, \widehat{\eta_j})\}_{j\in[m]} \subset {\mathbb{E}_L}^2$.
Output: A set of plaintext $\mathbf{Z} \subset \mathbb{M}$, such that $\mathbf{Z} = \mathbf{X} \cup \{\mathbf{L.D}_{sk_L}(\widehat{\eta_j})\mathbf{L.D}_{sk_L}(\widehat{e_j})^{-1} | \mathbf{L.D}_{sk_L} \neq 0\}$.

$\mathcal{R}$: compute $\mathbf{Z} \leftarrow \mathbf{X}$;
for all $j \in [m]$ **do**
    $\mathcal{R}$: compute $e_j \leftarrow \mathbf{L.D}_{sk_L}(\widehat{e_j})$;
    if $e_j \neq 0$ **then**
        $\mathcal{R}$: compute $\mathbf{Z} \leftarrow \mathbf{Z} + \{\mathbf{L.D}_{sk_L}(\widehat{\eta_j})e_j^{-1}\}$;
    end if
end for
$\mathcal{R}$: **return** $\mathbf{Z}$;

Proposition 3. *The protocol built with the algorithms* ***Setup***, ***Encode***, ***Reduce***, ***Map*** *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. □

Proposition 4. *For the receiver owning a set* $\boldsymbol{X}$ *of* n *elements, and the sender owning a set* $\boldsymbol{Y}$ *of* m *elements, with the assumption that* $n > m$, *the protocol built with the algorithms* ***Setup***, ***Encode***, ***Reduce***, ***Map*** *and* ***Union***, *respectively presented in Algs. 1 to 5, computes the set union with the asymptotic complexity bounds presented in Table 3.*

Table 3: Cost analysis of Protocol 1 for $n > m$

Algorithm	Ar. Cost for $\mathcal{R}$	Ar. Cost for $\mathcal{S}$	Comm. Vol.	Depth
Setup	$O(1)$	$O(1)$	$O(1)$	
Encode	$O(1)$	$O(m)$	$O(m)$	
Reduce	$O(mn)$	$O(1)$	$O(m)$	$\lceil \log n \rceil + 1$
Map	$O(1)$	$O(m)$	$O(m)$	
Union	$O(m)$	$O(1)$	$O(1)$	
Total	$O(mn)$	$O(m)$	$O(m)$	$\lceil \log n \rceil + 1$

Proof. The **Setup** algorithm requires a constant time and a constant communication volume as we can consider the scheme independent of the sizes of the party's sets. In **Encode**, $\mathcal{S}$ encrypts and batch m plaintexts, so $O(m)$ computations, and sends the batched ciphertext, which is equivalent to send m ciphertexts. In **Reduce**, the algorithm **F.BSMEv** dominates the cost and, accordingly to Prop. 1, it can be computed in $O(mn)$ arithmetic operations and a depth $\lceil \log n \rceil + 1$. One batched ciphertext, $\widetilde{\mathbf{h}}$, and the set of m ciphertexts, $\{\widehat{k_i}\}_{i\in[m]} \subset \mathbb{E}_L$ are exchanged, so $O(m)$ communications. In **Map**, the sender computes m FHE decryption and $O(m)$ LHE operations and $O(m)$ ciphertexts are exchanged. The receiver performs $O(m)$ LHE decryptions and $O(m)$ arithmetic operations on plaintexts in **Union**, so $O(m)$ operations. □

Overall, we have shown Theorem 1.

Theorem 1. *The protocol built with the algorithms **Setup**, **Encode**, **Reduce**, **Map** 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.*

Protocol 1: Optimal communication volume, low depth, batchable and parallelisable UPSU protocol

	$\mathcal{R}$ $\mathbf{X} = \{x_1, ..., x_n\}$		$\mathcal{S}$ $\mathbf{Y} = \{y_1, ..., y_m\}$	
Setup	$\{pk_F, sk_L, pk_L\}$	$\longleftrightarrow$	$\{pk_L, sk_F, pk_F\}$ $\{\widetilde{y_i}\}_{i\in[m]} \leftarrow \{\mathbf{F.E}_{pk_F}(y_i)\}_{i\in[m]}$	**Setup** **Encode**
Reduce	$P_{\mathcal{R}} \leftarrow \prod(Z - x_i)$	$\xleftarrow{\widetilde{\mathbf{y}}}$	$\widetilde{\mathbf{y}} \leftarrow \mathbf{F.Batch}((\widetilde{y_1} \quad \cdots \quad \widetilde{y_m}))$	
	$\{k_i\}_{i\in[m]} \xleftarrow{\$} \mathbb{M}_F$ $\{\widehat{k_i}\}_{i\in[m]} \leftarrow \{\mathbf{L.E}_{pk_L}(k_i)\}_{i\in[m]}$ $\{\widetilde{k_i}\}_{i\in[m]} \leftarrow \{\mathbf{F.E}_{pk_F}(k_i)\}_{i\in[m]}$ $\widetilde{\mathbf{k}} \leftarrow \mathbf{F.Batch}(\left(\widetilde{k_1} \quad \cdots \quad \widetilde{k_m}\right))$ $\widetilde{\mathbf{e}} \leftarrow \mathbf{F.BSMEv}(P_{\mathcal{R}}, \widetilde{\mathbf{y}}, \sqrt{n})$ $\widetilde{\mathbf{e}} \leftarrow \mathbf{F.BSMEv}(P_{\mathcal{R}}, \widetilde{\mathbf{y}}, \sqrt{n})$			
	$\widetilde{\mathbf{h}} \leftarrow \widetilde{\mathbf{k}} +_F \widetilde{\mathbf{e}}$	$\xrightarrow{\widetilde{\mathbf{h}},\ \{\widehat{k_i}\}_{i\in[m]}}$	$(h_1 \quad \cdots \quad h_m) \leftarrow \mathbf{F.D}_{sk_F}(\widetilde{\mathbf{h}})$	**Map**
			$\pi \xleftarrow{\$} \mathfrak{S}_m$ $\{\widehat{e_{\pi(i)}}\}_{i\in[m]} \leftarrow \{\widehat{h_i} -_L \widehat{k_i}\}_{i\in[m]}$	
Union	$\{e_j\}_{j\in[m]} \leftarrow \{\mathbf{L.D}_{sk_L}(\widehat{e_j})\}_{j\in[m]}$ $\begin{cases} e_j = 0 & \Rightarrow \quad \perp \\ e_j \neq 0 & \Rightarrow \quad \mathbf{X} \leftarrow \mathbf{X} + \{\mathbf{L.D}_{sk_L}(\widehat{\eta_j})\, e_j^{-1}\} \end{cases}$	$\xleftarrow{\{(\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}})\}_{i\in[m]}}$	$\{\widehat{\eta_{\pi(i)}}\}_{i\in[m]} \leftarrow \{y_i \ltimes_L \widehat{e_{\pi(i)}}\}_{i\in[m]}$	
	Return X			

4.4 Protocol timings in HElib

To instantiate the Protocol 1, we used the C++ open source library HElib which implements the BGV cryptosystem [Brakerski et al.(2014)]. This library is one of the most efficient that implements FHE schemes, in particular exact ones as BGV, and gives more control in the choice of the context parameters than other popular libraries as SEAL[6] and OpenFHE[7]. We used a 13th Gen Intel® Core™ i7-1370P with 20 threads and 32GB of RAM for our experiments. The plaintext space is the field $\mathbb{F}_{6143^{32}}$, so it can contains 384 bit-length words, and the context allows to batch up to 1024 ciphertexts together. In the following, we consider the size of the sender's set to be a constant $m = |\mathbf{Y}| = 1024$, and we want to analyze the communication volume and the runtime of both parties when the receiver's set size $n = |\mathbf{X}|$ grows exponentially, from 2^{10} to 2^{20} elements. For now, we only have implementations while using BGV restricted to linear operations as a LHE. In our application, the receiver is assumed to have an important computing power, so we parallelize as

[6] https://github.com/microsoft/SEAL
[7] https://github.com/openfheorg/openfhe-development

much a we can its computations, mainly **F.BSMEv** in **Reduce**, on the 20 available threads. We ignore the communication volume and the runtime implied by **Setup** as it can be done offline, so we consider that both parties own its secret key and both public keys. In HElib, we can (slightly) modify this maximum depth $d_{\max}$ by increasing the size of a fresh ciphertext. However, doing so reduces the security of the scheme. For that reason, we propose our results for different $d_{\max}$, starting from 21 to 11, with a computational security κ growing from 115 bits. Reducing the maximum depth also reduces the runtime of the protocol, but the reduction is anecdotic. In Fig. 1a, the experimental results confirm the expected asymptotic presented in Table 3, which are a cost for the receiver proportional to its set size. However, we can observe that the runtime for high values of n is huge, even while parallelizing. The Fig. 1b also confirms our expectations as the sender's runtime is independent of n, and this time the values obtained seems to be applicable in real life. In Fig. 1c, we can confirm that the communication volume of the protocol is independent of n for each scheme. However, by knowing the multiplicative depth generated by the protocol, which depends on n as the protocol has a depth $\log n + 1$, one can generate a context allowing exactly this depth as a maximum depth in order to reduce the communication volume. Note that in practice, it is the sender that generates the FHE scheme, so it might not know the value $\log n$, but just an upper bound on it. We also present in our graph the communication values claimed in [Tu et al.(2023)], but remember that in their implementations, the sender owns a set of 2^{10} 128-bit length elements, while our sender has 3 times this quantity of data. Also, we are not using the same fully homomorphic library nor scheme as they are using the SEAL implementation of FV [Fan and Vercauteren(2012)]. For $m = 2^{10}$ and $n = 2^{20}$, they claim to perform an UPSU under 7 seconds with 8 threads when we need more that 6400 seconds on 20 threads, but we are not using their optimizations (set hashing, partitioning, windowing, ...) even if we could, because it threaten the security [Kolesnikov et al.(2019)] and we want optimal communication volume asymptotic.

4.5 About the compatibility FHE-LHE

In Protocol 1 and in the following protocols of this paper, we made the assumption that the FHE and the LHE scheme can share the same plaintext space. If we are using schemes that does not satisfy that, for example BGV [Brakerski et al.(2014)] as a FHE, where the plaintext space can be a field $\mathbb{F}_{p^k}$, and Paillier [Yi et al.(2014)] as LHE, where the plaintext space is a ring $\mathbb{Z}_N$, the correctness of the protocol might be threaten. We show that it is possible to deal with those situations. The goal of our protocols is always to evaluate homomorphically a ciphertext polynomial in FHE plaintext evaluation points $\{y_i\}$, and for each i the evaluation point y_i should be retrieved if and only if it was not a root of the polynomial. More precisely, for each i, there are two FHE plaintexts $\alpha, \beta \in \mathbb{M}_F$ such that $\alpha = \beta$ if and only if $y_i \in \mathbb{M}_F$ is a root of the polynomial. The game is the following.

Game: Let $(\mathbf{L.Setup}, \mathbf{L.E}, \mathbf{L.D}, +_L, \ltimes_L)$ be an LHE scheme with plaintext space $\mathbb{M}_L$, $(pk_L, sk_L) \leftarrow \mathbf{L.Setup}(\kappa)$ be the scheme keys for a security parameter κ and let $\mathbb{M}_F$ be a FHE plaintext space. Suppose that there is a one-to-one correspondence $\Psi : \mathbb{M}_F \longrightarrow \mathrm{Im}(\Psi) \subset {\mathbb{M}_L}^\lambda$ such that $\Psi(0) = 0$. Let $y, \alpha, \beta \in \mathbb{M}_F$ be three FHE plaintexts and $P \in \mathbb{M}_F[Z]$ be a plaintext polynomial such that

$$\alpha = \beta \iff P(y) = 0. \tag{31}$$

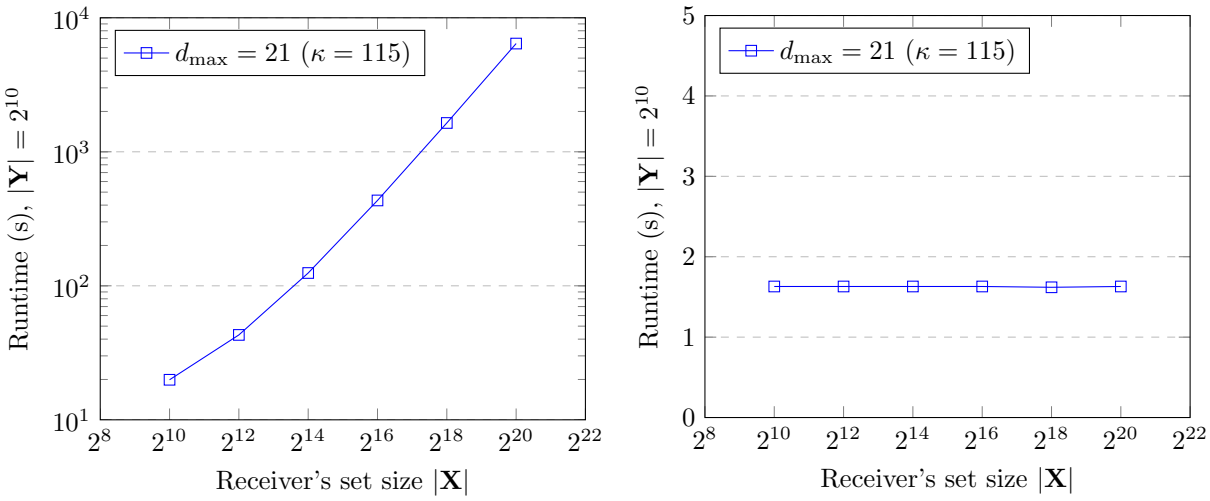
There are two players, $\mathcal{P}_1$ and $\mathcal{P}_2$. $\mathcal{P}_1$ owns $\{y, \widehat{\alpha}, \widehat{\beta}\}$, where $\widehat{\alpha} \leftarrow \mathbf{L.E}_{pk_L}(\Psi(\alpha))$ (similarly for $\widehat{\beta}$), and $\mathcal{P}_2$ owns $\{sk_L\}$. With some communication between $\mathcal{P}_1$ and $\mathcal{P}_2$, the goal is to allow $\mathcal{P}_2$ to compute y if and only if $P(y) \neq 0$ without revealing anything about P, α or β to $\mathcal{P}_1$ From this point, we will consider three situations on the schemes that lead to three ways to solve the game.

Situation 1: $\mathbb{M}_L = \mathbb{M}_F$ and it is a field.
This is the easiest situation because there is no compatibility issue between the LHE and the FHE scheme;

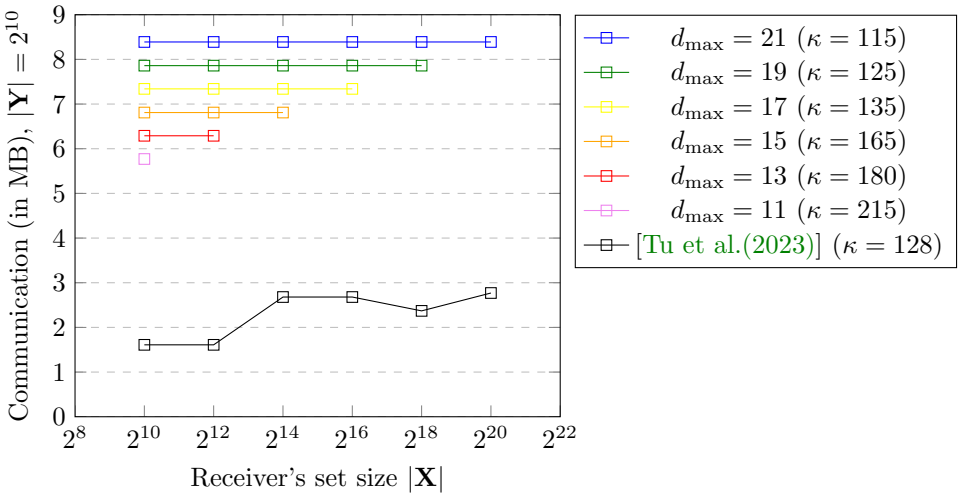
Figure 1: Protocol 1 experimental results

(a) Receiver's runtime of Protocol 1

(b) Sender's runtime of Protocol 1



(c) Communication volume of Protocol 1



the map Ψ is the identity. We present all our protocols in that situation for sake of clarity, and it is always possible to restrict a FHE scheme for which the plaintext space is a field to LHE operations. To win the game, as seen in our protocols, the player $\mathcal{P}_1$ computes

$$\widehat{e} \leftarrow \widehat{\alpha} -_L \widehat{\beta}, \tag{32}$$

$$\widehat{\eta} \leftarrow y \ltimes_L \widehat{e}. \tag{33}$$

and sends $\{\widehat{e}, \widehat{\eta}\}$ to $\mathcal{P}_2$. Using its secret key, $\mathcal{P}_2$ decrypts

$$e \leftarrow \mathbf{L.D}_{sk_L}(\widehat{e}), \tag{34}$$

$$\eta \leftarrow \mathbf{L.D}_{sk_L}(\widehat{\eta}). \tag{35}$$

Then, if $P(y) = 0$, so $\alpha = \beta$, the correctness of the LHE scheme implies that $e = \eta = 0$, and if $P(y) \neq 0$, $y = \eta e^{-1}$.

Situation 2: $\mathbb{M}_L \simeq \mathbb{Z}_N$, $\mathbb{M}_F \simeq \mathbb{F}_q$, with $q = p^k$ for p prime, and $q^2 < N$.
In practice, the usage of FHE scheme, even restricted to the LHE functionalities, is more expensive than using a pure LHE; for that reason, we would rather use a pure LHE, but the correctness may be threaten. Remember that $\mathbb{F}_q = \mathbb{F}_{p^k} \simeq \frac{\mathbb{F}_p[X]}{(T(X))}$ for $T(X)$ a polynomial of degree k, irreducible on $\mathbb{F}_p$. Then, for a chosen $T(X)$, an element $\alpha \in \mathbb{F}_q$ is uniquely represented as a polynomial $\sum_{i=0}^{k-1} a_i X^i \in \frac{\mathbb{F}_p[X]}{(T(X))}$, where $a_i \in \mathbb{F}_p$ for all i. By viewing this polynomial as a polynomial in $\mathbb{Z}[X]$, we introduce the one-to-one correspondence Ψ between $\mathbb{F}_q$ and $[0, q)$ that maps $\alpha \in \mathbb{F}_q$ to $\sum_{i=0}^{k-1} a_i p^i \in [0, q)$. With the condition, an element $\alpha \in \mathbb{F}_q$ is uniquely represented in $\mathbb{Z}_N$ with $\Psi(\alpha) \in [0, q) \subset [0, N)$. Note that $\Psi(0) = 0$ and Ψ does not conserve the arithmetic. In particular, for $\alpha, \beta \in \mathbb{F}_q$, $\Psi(\alpha + \beta)$ may not be equal to $\Psi(\alpha) + \Psi(\beta)$ in $\mathbb{Z}_N$, and $\Psi(\alpha\beta)$ may differ from $\Psi(\alpha)\Psi(\beta)$. To win the game, the player $\mathcal{P}_1$ computes

$$\widehat{e} \leftarrow \widehat{\alpha} -_L \widehat{\beta}, \tag{36}$$
$$\widehat{\eta} \leftarrow \Psi(y) \ltimes_L \widehat{e}, \tag{37}$$
$$\widehat{\nu} \leftarrow \Psi(y) \ltimes_L (\widehat{0} -_L \widehat{e}), \tag{38}$$

for $\widehat{0}$ an encryption of 0, and sends $\{\widehat{e}, \widehat{\eta}, \widehat{\nu}\}$ to $\mathcal{P}_2$. Using its secret key, $\mathcal{P}_2$ decrypts

$$e \leftarrow \mathbf{L.D}_{sk_L}(\widehat{e}), \tag{39}$$
$$\eta \leftarrow \mathbf{L.D}_{sk_L}(\widehat{\eta}), \tag{40}$$
$$\nu \leftarrow \mathbf{L.D}_{sk_L}(\widehat{\nu}). \tag{41}$$

The correctness of the LHE scheme assures that

$$e = \Psi(\alpha) - \Psi(\beta) \mod N, \tag{42}$$
$$\eta = \Psi(y)e \mod N, \tag{43}$$
$$\nu = -\Psi(y)e \mod N. \tag{44}$$

First, if $\alpha = \beta$, both e, η and ν are 0. Now, if $\alpha \neq \beta$, there are two situations, depending if a modular reduction has been done in e when subtracting $\Psi(\alpha) \in [0, q)$ to $\Psi(\beta) \in [0, q)$. If $e \in [0, q)$, then $\eta \in [0, q^2)$ and with an inversion in the rational, we have

$$\Psi(y) = \Psi(y)ee^{-1} \in \mathbb{Q}. \tag{45}$$

If now $e \in [N - q, N)$, then $\nu = \Psi(y)(N - e) \in [0, q^2)$ and with an inversion in the rational, we have

$$\Psi(y) = \Psi(y)(N - e)(N - e)^{-1} \in \mathbb{Q}. \tag{46}$$

Finally, if and only if $P(y) \neq 0$, so $\alpha \neq \beta$, $\mathcal{P}_2$ can compute back

$$y = \Psi^{-1}(\Psi(y)) \in \mathbb{F}_q. \tag{47}$$

Situation 3: Other cases.
If we are not in the previous situations, we can always use an oblivious transfer [Naor and Pinkas(2001)] to let $\mathcal{P}_2$ ask obliviously y if and only if $\alpha \neq \beta$. This solution is not optimal in terms of number of rounds. It explains why we don't focus much on it. To win the game, the player $\mathcal{P}_1$ computes

$$\widehat{e} \leftarrow \widehat{\alpha} - \widehat{\beta}, \tag{48}$$

and sends $\widehat{e}$ to $\mathcal{P}_2$. Using the secret key, $\mathcal{P}_2$ decrypts

$$e \leftarrow \mathbf{L.D}_{sk_L}(\widehat{e}), \tag{49}$$

and fix the bit b to 1 if and only if $e = 0$. Then, the two parties invoke an oblivious transfer protocol where $\mathcal{P}_1$ acts as the sender and inputs $(m_0, m_1) := (y, \perp)$, while $\mathcal{P}_2$ acts as the receiver, inputs the choice bit b and outputs m_b.

5 Optimal communication UPSU : theoretical variants using efficient homomorphic polynomial algorithm

5.1 Notations

In this section, we will use the following notations. Let $A = \sum_{i=0}^{d} a_i Z^i$ and $B = \sum_{i=0}^{d'} b_i Z^i$, with $d \leq d'$, be two polynomials in $\mathbb{M}[Z]$. The *reverse polynomial* of A is defined to be

$$\overleftarrow{A} := \sum_{i=0}^{d} a_{d-i} Z^i = A(\frac{1}{Z}) Z^d. \tag{50}$$

For $0 \leq l \leq t \leq d$, let

$$[A]_l^t := \sum_{i=l}^{t} a_i Z^{i-l}. \tag{51}$$

The *middle product* of A by B is defined to be

$$[\overleftarrow{A} B]_d^{d'} = \sum_{i=0}^{d'-d} \sum_{j=0}^{d} a_j b_{j+i} Z^i. \tag{52}$$

As often in computer algebra, we will reduce the arithmetic cost of our algorithms to the number of polynomial products done. However, using homomophic encryption implies that such polynomial product are not done in the same context, so in particular the algorithm that instantiates the product of two ciphertext polynomials in an FHE scheme, the one that instantiates the product of one plaintext and one ciphertext polynomial in an LHE scheme and the one that multiplies two plaintext polynomials may differ, so may have different asymptotics. For that reason, and in order not to have to instantiate the homomorphic schemes, we will denote the arithmetic costs of such polynomial products as following.

- $\mathcal{M}(d)$ is a bound on the arithmetic cost of product between two polynomials of degrees at most d.
- $\mathcal{M}_F(d)$ is a bound on the arithmetic cost of the map $\widetilde{A}, \widetilde{B} \mapsto \widetilde{A} \times_F \widetilde{B}$ for A, B of degrees at most d. A homomorphic polynomial product has depth 1.
- $\mathcal{M}_L(d)$ is a bound on the arithmetic cost of the map $A, \widehat{B} \mapsto A \ltimes_L \widehat{B}$ for A, B of degrees at most d.

5.2 Efficient protocol when $\sqrt{n} < m < n$ using fully homomorphic euclidean remainder and linearly homomorphic multi-point evaluation

In our first UPSU variant, presented in Protocol 2, we have to suppose that we are in the **Situation 1** from Section 4.5, which means that the plaintext space of the LHE scheme $\mathbb{M}_L$ and the plaintext space of the FHE scheme $\mathbb{M}_F$ are the same field, that we will denote $\mathbb{M}$. The following protocol is based on efficient polynomial algorithms adapted in the homomorphic context, namely a homomorphic euclidean remainder under FHE encryption and an efficient multi-point evaluation under LHE encryption. Overall, the following protocol has a better asymptotical arithmetic cost for the receiver than Protocol 1, but it increases slightly the sender's cost. Also, if we consider the size of the sender's set m to satisfy $\sqrt{n} < m < n$, where n is the receiver's set size, this protocol has a lower multiplicative depth than the previous protocol.

5.2.1 Efficient linearly homomorphic multi-point evaluation

The first building block is a linearly homomorphic multi-point polynomial evaluation algorithm, denoted **L.MEv**, that evaluates homomorphically a ciphertext polynomial of degree $m-1$ in m plaintext evaluation points.

Definition 5.1. Let $(\mathbf{L.Setup}, \mathbf{L.E}, \mathbf{L.D}, +_L, \ltimes_L)$ be a LHE scheme. The linearly homomorphic multi-point evaluation is an algorithm **L.MEv** such that

- $\{\widehat{e_1}, \ldots, \widehat{e_m}\} \leftarrow \mathbf{L.MEv}(\widehat{H}, \{y_1, \ldots, y_m\})$: Given as inputs an encrypted polynomial $\widehat{H} \in \mathbb{E}_L[Z]$ and a set of m plaintexts $y_1, \ldots, y_m \in \mathbb{M}$, outputs a set of m ciphertexts $\widehat{e_1}, \ldots, \widehat{e_m} \in \mathbb{E}_L$.

The algorithm satisfies the following correctness property.

Correctness. For a security parameter κ, for every $H \in \mathbb{M}[Z]$, every subset $\{y_1, \ldots, y_m\} \subset \mathbb{M}$ and $(pk, sk) \leftarrow \mathbf{L.Setup}(\kappa)$, if

$$\{\widehat{e_1}, \ldots, \widehat{e_m}\} \leftarrow \mathbf{L.MEv}\left(\mathbf{L.E}_{pk}\left(H\right), \{y_1, \ldots, y_m\}\right) \tag{53}$$

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

$$\mathbf{L.D}_{sk}(\widehat{e_i}) = H(y_i). \tag{54}$$

Naively, the algorithm **L.MEv** can be implemented in $O(m^2)$ operations, evaluating homomorphically the polynomial on each point with an adaptation of the Horner scheme. There exist an asymptotically fast multi-point evaluation algorithm described in [Bostan et al.(2003)]. The main idea is to use the transposition principle that guarantees that an algorithm computing a linear application can be transposed in an algorithm computing the transposed linear application in about the same computation time. There is an efficient algorithm computing the transposed linear application of the multi-point evaluation, using polynomial products, so the transposition principle gives an efficient multi-point evaluation. With the notations of Rem. 2, we were able to adapt this algorithm to the LHE setting.

Proposition 5. *Let $\widehat{H} \in \mathbb{E}_L[Z]$ be a LHE ciphertext polynomial of degree $m-1$ and $y_1, \ldots, y_m \in \mathbb{M}$ be m plaintext evaluation points.*
L.MEv$(\widehat{H}, \{y_1, \ldots, y_m\})$ *can be computed in $\mathcal{M}_L(m)\log m + \widetilde{O}(m)$ operations, after $\frac{1}{2}\mathcal{M}(m)\log m + \widetilde{O}(m)$ operations of precomputation on $y_1, \ldots, y_m$.*

Proof. We adapt the algorithm presented in [Bostan et al.(2003)] to the LHE context. Let $\widehat{H} = \sum_{i=0}^{m-1} \widehat{h_i} Z^i$, $H \leftarrow \mathbf{L.D}_{sk}(\widehat{H})$. and $y_1, \ldots, y_m \in \mathbb{M}$. We assume that m 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 $k = 0, \ldots, \log m$ and $i = 1 \ldots, 2^k$:

$$P_{\left(\frac{i}{2^k}\right)} := \prod_{j \in \left\{\frac{i-1}{2^k}m+1, \ldots, \frac{i}{2^k}m\right\}} (Z - y_j) \tag{55}$$

These polynomials can be computed using a product tree in $\frac{1}{2}\mathcal{M}(m)\log m + \widetilde{O}(m)$ 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

$$B := \overleftarrow{P_{\left(\frac{1}{1}\right)}}^{-1} \mod Z^m, \text{ and} \tag{56}$$

$$\widehat{A} := \left[\overleftarrow{B} \ltimes_L \widehat{H}\right]_{m-1}^{2m-1}. \tag{57}$$

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

$$\widehat{A}_{\left(\frac{i}{2^k}\right)} = \mathbf{L.Mid}\left(P_{\left(\frac{i-(-1)^{(i \mod 2)}}{2^k}\right)}, \widehat{A}_{\left(\frac{\lceil i/2 \rceil}{2^{k-1}}\right)}\right). \tag{58}$$

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

5.2.2 Efficient fully homomorphic euclidean remainder

The second building block is the homomorphic computation of a polynomial remainder. In the context of homomorphic encryption, the impossibility of branching, for instance, makes this a non-trivial task. For our purpose, 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 thus focus here on the case where the divisor is monic. We denote this algorithm with the operator $\mathbf{mod}_F$.

Definition 5.2. 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 operator $\mathbf{mod}_F$ as follows:

- $\widetilde{R} \leftarrow A \ \mathbf{mod}_F \left(\widetilde{B}\right)$: Given as inputs a plaintext polynomial $A \in \mathbb{M}[Z]$ and an encrypted polynomial $\widetilde{B} \in \mathbb{E}_F[Z]$, outputs an encrypted polynomial $\widetilde{R} \in \mathbb{E}_F[Z]$.

This algorithm satisfies the following correctness property.

Correctness. For a security parameter κ, for $(pk, sk) \leftarrow \mathbf{F.Setup}(\kappa)$, and for two plaintext polynomials A, $B \in \mathbb{M}[Z]$ such that B is monic,

$$\mathbf{F.D}_{sk}\left(A \ \mathbf{mod}_F \left(\mathbf{F.E}_{pk}\left(B\right)\right)\right) = A \bmod B. \tag{59}$$

The standard algorithm for this task is the quadratic long division algorithm. In our algorithm, we adapt the fast euclidean division algorithm, based on Newton iteration (using only polynomial addition and product)[von zur Gathen and Gerhard(2013)], to the FHE settings. The algorithm is quasi-linear and has only a logarithmic multiplicative depth.

Proposition 6. *Let $A \in \mathbb{M}[Z]$ be a plaintext polynomial of degree n and let $\widetilde{B} \in \mathbb{E}_F[Z]$ be a FHE ciphertext polynomial which is an encryption of a monic polynomial of degree $m < n$. A $\boldsymbol{mod}_F \left(\widetilde{B}\right)$ can be computed in less than $4\mathcal{M}_F(n-m) + O(n)$ arithmetic operations with a depth $2\lceil \log(n-m+1) \rceil + 1$.*

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 a monic B, of respective degrees n and $m < n$, is the unique polynomial satisfying $A = BQ + R$ with $\deg(R) < m$. This implies $\overleftarrow{A} = \overleftarrow{Q}\overleftarrow{B} + Z^{n-m+1}\overleftarrow{R}$, whence

$$\overleftarrow{Q} = \overleftarrow{A}\,\overleftarrow{B}^{-1} \mod Z^{n-m+1}. \tag{60}$$

The goal is to homomorphically compute the inverse of $\overleftarrow{B}$ modulo Z^{n-m+1}, using Newton iteration. Let $\widetilde{A}$ and $\widetilde{B}$ be 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 coefficient $\widetilde{U_L}$, for $L = \lceil \log(n-m+1) \rceil - 1$, of the sequence $(\widetilde{U})$:

$$(\widetilde{U}) = \begin{cases} \widetilde{U_0} = \widetilde{1} \\ \widetilde{U_{k+1}} = \widetilde{U_k} \times_F \left(\widetilde{1} -_F \left[\overleftarrow{\widetilde{B}} \times_F \widetilde{U_k} \right]_{2^k}^{2^{k+1}-1} Z^{2^k} \right) \mod Z^{2^{k+1}} \end{cases} \tag{61}$$

Now, instead of computing the last step of the sequence that would give us homomorphically the inverse polynomial of $\overleftarrow{B} \mod Z^{n-m+1}$, we directly compute homomorphically the quotient, using $\widetilde{U_L}$.

$$\widetilde{S} = A \ltimes_F \widetilde{U}_L \mod Z^{n-m+1}, \text{ and} \tag{62}$$

$$\widetilde{T} = \left[\overleftarrow{\widetilde{B}} \times_F \widetilde{U}_L \right]_{2^L}^{2^{L+1}-1} \times_F \left[\widetilde{S} \right]_0^{n-m-2^L} \mod Z^{n-m+1-2^L}. \tag{63}$$

Then $\overleftarrow{\widetilde{Q}} := \widetilde{S} +_F \widetilde{T} Z^{2^L}$ is an encryption of $\overleftarrow{Q}$, the reverse quotient. Finally, we compute

$$\widetilde{R} = A -_F \widetilde{Q} \times_F \widetilde{B} \mod Z^m \tag{64}$$

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 at most $4\mathcal{M}_F(n-m) + O(n)$. Also, each step of the sequence $(\widetilde{U})$ requires one polynomial product and one middle product (homomorphically) which means that computing $\widetilde{U_L}$ has depth $2L$. To obtain $\widetilde{R}$, it requires 3 more products, so in total, this algorithm has a depth $2\lceil \log(n-m+1) \rceil + 1$. □

Remark 6. As mentioned in the previous proof, and by using the same notations, each step of the sequence $(\widetilde{U})$ increases the depth by two. Starting with $\widetilde{U_0} = \widetilde{1}$ leads to a total depth of $2\lceil \log(n-m+1) \rceil + 1$ to compute $A \bmod_F \left(\widetilde{B} \right)$. However, if one starts the sequence $(\widetilde{U})$ with the ciphertext polynomial $\widetilde{U_l}$ that is an encryption of $\overleftarrow{B}^{-1} \mod Z^{2^l}$, for $l := \lceil \log m \rceil$, computing $A \bmod_F \left(\widetilde{B} \right)$ has now a depth $2\left(\lceil \log(n-m+1) \rceil - \lceil \log m \rceil \right) + 1$ (the arithmetic cost remains unchanged as the last steps are the most expensive).

5.2.3 Protocol with $\bmod_F$ and L.MEv

Formally, our protocol is built with the algorithms **Setup**, **Encode**, **Reduce**, **Map** and **Union** respectively presented in Algs. 1 and 5 to 8. A more visual version is presented in Protocol 2.

Theorem 2. *The protocol built with the algorithms **Setup**, **Encode**, **Reduce**, **Map** and **Union**, respectively presented in Algs. 1 and 5 to 8, is a secure unbalanced private set union scheme (UPSU) under the honest-but-curious adversary model. For the receiver owning a set $\boldsymbol{X}$ of n plaintexts, and the sender owning a set $\boldsymbol{Y}$ of m plaintexts, with the assumption that $n > m$, it computes the set union with the asymptotic complexity bounds presented in Table 4.*

Proof. The complete simulation proof is presented in Appendix A.2. The asymptotic presented in Table 4 are implied by Prop. 5 and Prop. 6 (together with the analysis of Rem. 6). We assume that the homomorphic operations are correct and we use the same notations than in Protocol 2. We are in the **Situation 1** from Section 4.5, so everything works properly while switching from FHE to LHE scheme. In **Reduce**, $\widetilde{R}$ is an encryption of the remainder R of $P_{\mathcal{R}}$ divided by $P_{\mathcal{S}}$. In particular, for all $y \in \mathbf{Y}$, $R(y) = 0 \Leftrightarrow y \in \mathbf{X}$. In **Map**, for all $i \in [m]$, $h_i = R(y_i) + M(y_i)$ and $\widehat{m_i}$ is an encryption of $M(y_i)$. In implies that $\widehat{e_{\pi(i)}}$ is an encryption of $R(y_i)$. Finally, in **Union**, for all $j \in [m]$, $e_j = 0 \Leftrightarrow R(y_{\pi^{-1}(j)}) = 0 \Leftrightarrow y_{\pi^{-1}(j)} \in \mathbf{X}$, and for the $e_j \neq 0$, as $\eta_j = e_j y_{\pi^{-1}(j)}$ and we are in a field, $y_{\pi^{-1}(j)} = \eta_j e_j^{-1}$. □

Algorithm 6 Encode$(\mathbf{Y}, keys_{\mathcal{S}})$

Input: A set of plaintexts $\mathbf{Y} = \{y_i\}_{i\in[m]} \subset \mathbb{M}$ and $keys_{\mathcal{S}} = \{(pk_F, sk_F), pk_L\}$.
Output: Two encrypted polynomials $\widetilde{P_{\mathcal{S}}} \in \mathbb{E}_F[Z]$ and $\widetilde{U_l} \in \mathbb{E}_F[Z]$ such that $\mathbf{F.D}_{sk_F}(\widetilde{P_{\mathcal{S}}}) = \prod_{i\in[m]} (Z - y_i)$ and $\mathbf{F.D}_{sk_F}(\widetilde{U_l}) = \overleftarrow{\prod_{i\in[m]} (Z - y_i)}^{-1} \mod Z^{2^{\lceil \log m \rceil}}$.

1: $\mathcal{S}$: compute $P_{\mathcal{S}} \leftarrow \prod_{y\in\mathbf{Y}} (Z - y)$;
2: $\mathcal{S}$: compute $U_l \leftarrow \overleftarrow{P_{\mathcal{S}}}^{-1} \mod Z^{2^{\lceil \log m \rceil}}$;
3: $\mathcal{S}$: compute $\widetilde{P_{\mathcal{S}}} \leftarrow \mathbf{F.E}_{pk_F}(P_{\mathcal{S}})$ and $\widetilde{U_l} \leftarrow \mathbf{F.E}_{pk_F}(P_{\mathcal{S}})$, and send $\widetilde{P_{\mathcal{S}}}, \widetilde{U_l}$ to $\mathcal{R}$;
4: $\mathcal{R}$: **return** $\{\widetilde{P_{\mathcal{S}}}, \widetilde{U_l}\}$;

Algorithm 7 Reduce$(\mathbf{X}, \{\widetilde{P_{\mathcal{S}}}, \widetilde{U_l}\}, keys_{\mathcal{R}})$

Input: A set of plaintext $\mathbf{X} = \{x_j\}_{j\in[n]} \subset \mathbb{M}$, two ciphertext polynomials $\{\widetilde{P_{\mathcal{S}}}, \widetilde{U_l}\} \subset \mathbb{E}_F[Z]$ and $keys_{\mathcal{R}} = \{(pk_L, sk_L), pk_F\}$.
Output: Two encrypted polynomials $\widetilde{H} \in \mathbb{E}_F[Z]$ and $\widehat{M} \in \mathbb{E}_L[Z]$, such that $\mathbf{F.D}_{sk_F}(\widetilde{H}) - \mathbf{L.D}_{sk_L}(\widehat{M}) = \prod_{j\in[n]} (Z - x_j) \mod \mathbf{F.D}_{sk_F}(\widetilde{P_{\mathcal{S}}})$.

1: $\mathcal{R}$: compute $P_{\mathcal{R}} \leftarrow \prod_{x\in\mathbf{X}} (Z - x)$;
2: $\mathcal{R}$: randomly select $M \xleftarrow{\$} \mathbb{M}[Z]$ such that $\deg(M) = \deg(\widetilde{P_{\mathcal{S}}}) - 1$;
3: $\mathcal{R}$: compute $\widetilde{M} \leftarrow \mathbf{F.E}_{pk_F}(M)$;
4: $\mathcal{R}$: compute $\widehat{M} \leftarrow \mathbf{L.E}_{pk_L}(M)$;
5: $\mathcal{R}$: compute $\widetilde{R} \leftarrow P_{\mathcal{R}} \ \mathbf{mod}_F \left(\widetilde{P_{\mathcal{S}}}\right)$; ▷ Using $\widetilde{U_l}$ to reduce the multiplicative depth (cf. Rem. 6).
6: $\mathcal{R}$: compute $\widetilde{H} \leftarrow \widetilde{R} +_F \widetilde{M}$;
7: $\mathcal{R}$: compute $\widetilde{H} \leftarrow \mathbf{low}(\widetilde{H})$;
8: $\mathcal{R}$: send $\{\widetilde{H}, \widehat{M}\}$ to $\mathcal{S}$;
9: $\mathcal{S}$: **return** $\{\widetilde{H}, \widehat{M}\}$;

Table 4: Cost analysis of Protocol 2 for $n > m$

Algorithm	Ar. Cost for $\mathcal{R}$	Ar. Cost for $\mathcal{S}$	Comm. Vol.	Depth
Setup	$O(1)$	$O(1)$	$O(1)$	
Encode	$O(1)$	$\widetilde{O}(m)$	$O(m)$	
Reduce	$4\mathcal{M}_F(n) + O(n)$	$O(1)$	$O(m)$	$2(\lceil\log(n-m+1)\rceil - \lceil\log m\rceil) + 1$
Map	$O(1)$	$\mathcal{M}_L(m)\log m + \widetilde{O}(m)$	$O(m)$	
Union	$O(m)$	$O(1)$	$O(1)$	
Total	$4\mathcal{M}_F(n) + O(n)$	$\mathcal{M}_L(m)\log m + \widetilde{O}(m)$	$O(m)$	$2(\lceil\log(n-m+1)\rceil - \lceil\log m\rceil) + 1$

5.3 Optimal communication and arithmetic cost protocol with compatibility LHE-FHE

The second theoretical variant proposed is adaptable in both **Situations** of Section 4.5; for sake of clarity, only the **Situation 1** from Section 4.5 is described visually in Protocol 3. This one is optimal in terms of communication volume and (quasi) optimal for the sender's arithmetic cost, has the same asymptotic for the receiver's arithmetic cost than Protocol 2 , but the multiplicative depth is doubled. It is constructed with an efficient multi-point evaluation based on successive euclidean remainder adapted in the FHE context.

Algorithm 8 $\mathbf{Map}(\mathbf{Y}, \{\widetilde{H}, \widehat{M}\}, keys_{\mathcal{S}})$

Input: A set of m plaintexts $\mathbf{Y} = \{y_i\}_{i\in[m]} \subset \mathbb{M}$, two ciphertext polynomials $\{\widetilde{H}, \widehat{M}\}$, resp. in $\mathbb{E}_F[Z]$ and $\mathbb{E}_L[Z]$, and $keys_{\mathcal{S}} = \{(pk_F, sk_F), pk_L\}$.
Output: A set of ciphertext tuples $\{(\widehat{e_j}, \widehat{\eta_j})\}_{j\in[m]} \subset {\mathbb{E}_L}^2$, such that for all $i \in [m]$, $\mathbf{L.D}_{sk_L}(\widehat{e_{\pi(i)}}) = \left(\mathbf{F.D}_{sk_F}(\widetilde{H}) - \mathbf{L.D}_{sk_L}(\widehat{M})\right)(y_i)$ and $\mathbf{L.D}_{sk_L}(\widehat{\eta_{\pi(i)}}) = y_i \mathbf{L.D}_{sk_L}(\widehat{e_{\pi(i)}})$.

1: $\mathcal{S}$: compute $H \leftarrow \mathbf{F.D}_{sk_F}(\widetilde{H})$;
2: $\mathcal{S}$: compute $\{h_i\}_{i\in[m]} \leftarrow \mathbf{MEv}(H, \mathbf{Y})$;
3: $\mathcal{S}$: compute $\{\widehat{m_i}\}_{i\in[m]} \leftarrow \mathbf{L.MEv}(\widehat{M}, \mathbf{Y})$;
4: $\mathcal{S}$: randomly select $\pi \xleftarrow{\$} \mathfrak{S}_m$;
5: **for all** $i \in [m]$ **do**
6: $\quad\mathcal{S}$: compute $\widehat{e_{\pi(i)}} \leftarrow \mathbf{L.E}_{pk_L}(h_i) -_L \widehat{m_i}$;
7: $\quad\mathcal{S}$: compute $\widehat{\eta_{\pi(i)}} \leftarrow y_i \ltimes_L \widehat{e_{\pi(i)}}$;
8: **end for**
9: $\mathcal{S}$: send $\{(\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}})\}_{i\in[m]}$ to $\mathcal{R}$;
10: $\mathcal{R}$: **return** $\{(\widehat{e_j}, \widehat{\eta_j})\}_{j\in[m]}$;

Protocol 2: Optimal communication volume UPSU protocol with $\mathbf{mod}_F$ and $\mathbf{L.MEv}$

	$\mathcal{R}$		$\mathcal{S}$	
	$\mathbf{X} = \{x_1, ..., x_n\}$		$\mathbf{Y} = \{y_1, ..., y_m\}$	
Setup	$\{pk_F, sk_L, pk_L\}$	$\longleftrightarrow$	$\{pk_L, sk_F, pk_F\}$	**Setup**
	$P_{\mathcal{R}} \leftarrow \prod(Z - x_i)$		$P_{\mathcal{S}} \leftarrow \prod(Z - y_i)$	
			$U_l \leftarrow \overleftarrow{P_{\mathcal{S}}}^{-1} \mod Z^{2^{\lceil \log m \rceil}}$	
Reduce	$M \xleftarrow{\$} \mathbb{M}[Z]_{m-1}$	$\xleftarrow{\widetilde{P_{\mathcal{S}}}, \widetilde{U_l}}$	$\widetilde{P_{\mathcal{S}}}, \widetilde{U_l} \leftarrow \mathbf{F.E}_{pk_F}(P_{\mathcal{S}}, U_l)$	**Encode**
	$\widehat{M} \leftarrow \mathbf{L.E}_{pk_L}(M)$			
	$\widetilde{M} \leftarrow \mathbf{F.E}_{pk_F}(M)$			
	$\widetilde{R} \leftarrow P_{\mathcal{R}} \ \mathbf{mod}_F \left(\widetilde{P_{\mathcal{S}}}\right)$			
	$\widetilde{H} \leftarrow \widetilde{R} +_F \widetilde{M}$	$\xrightarrow{\widetilde{H},\ \widehat{M}}$	$H \leftarrow \mathbf{F.D}_{sk_F}(\widetilde{H})$	**Map**
			$\{h_i\}_{i\in[m]} \leftarrow \mathbf{MEv}(H, \mathbf{Y})$	
			$\{\widehat{m_i}\}_{i\in[m]} \leftarrow \mathbf{L.MEv}(\widehat{M}, \mathbf{Y})$	
			$\pi \xleftarrow{\$} \mathfrak{S}_m$	
			$\{\widehat{e_{\pi(i)}}\}_{i\in[m]} \leftarrow \{\widehat{h_i} -_L \widehat{m_i}\}_{i\in[m]}$	
Union	$\{e_j\}_{j\in[m]} \leftarrow \{\mathbf{L.D}_{sk_L}(\widehat{e_j})\}_{j\in[m]}$	$\xleftarrow{\{(\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}})\}_{i\in[m]}}$	$\{\widehat{\eta_{\pi(i)}}\}_{i\in[m]} \leftarrow \{y_i \ltimes_L \widehat{e_{\pi(i)}}\}_{i\in[m]}$	
	$\begin{cases} e_j = 0 & \Rightarrow \quad \perp \\ e_j \neq 0 & \Rightarrow \quad \mathbf{X} \leftarrow \mathbf{X} + \{\mathbf{L.D}_{sk_L}(\widehat{\eta_j})\, e_j^{-1}\} \end{cases}$			
	Return X			

5.3.1 Efficient fully homomorphic multi-point evaluation

The main building block of the Protocol 3 is an efficient fully homomorphic multi-point evaluation algorithm, denoted **F.MEv**, that evaluates homomorphically a plaintext polynomial in encrypted evaluation points.

Definition 5.3. Let $(\mathbf{F.Setup}, \mathbf{F.E}, \mathbf{F.D}, +_F, \ltimes_F, \times_F)$ be a FHE scheme. The fully homomorphic multi-point evaluation is an algorithm **F.MEv** such that

- $\{\widetilde{e_1}, ..., \widetilde{e_m}\} \leftarrow \mathbf{F.MEv}(A, \{\widetilde{y_1}, ...\widetilde{y_m}\})$: Given as inputs a plaintext polynomial $A \in \mathbb{M}_F[Z]$, a set of m ciphertexts $\{\widetilde{y_1}, ...\widetilde{y_m}\} \subset \mathbb{E}_F$, outputs a set of m ciphertexts $\{\widetilde{e_1}, ..., \widetilde{e_m}\} \subset \mathbb{E}_F$.

This algorithm satisfies the following correctness property.

Correctness. For a security parameter κ, for $(pk, sk) \leftarrow \mathbf{F.Setup}(\kappa)$, for a plaintext polynomials $A \in \mathbb{M}_F[Z]$ and for m plaintext evaluation points $\{y_1, ..., y_m\} \subset \mathbb{M}_F$, if

$$\{\widetilde{e_1}, ..., \widetilde{e_m}\} \leftarrow \mathbf{F.MEv}(A, \{\mathbf{F.E}_{pk}(y_1), ...\mathbf{F.E}_{pk}(y_m)\}) \tag{65}$$

then, for all $i \in \{1, ..., m\}$,

$$\mathbf{F.D}_{sk}(\widetilde{e_i}) = A(y_i) \tag{66}$$

With an adaptation of the Newton iteration and an multi-point algorithm based on successive euclidean remainders, adapted in the fully homomorphic context, we are able to prove the following proposition.

Proposition 7. *Let $A \in \mathbb{M}_F[Z]$ be of degree n and $\{\widetilde{y_1}, ..., \widetilde{y_m}\} \subset \mathbb{E}_F$ be $m < n$ ciphertexts. $\boldsymbol{F.MEv}(A, \{\widetilde{y_1}, ...\widetilde{y_m}\})$ can be computed in less than $4\mathcal{M}_F(n-m)+O(n)$ arithmetic operations with a depth $2(L+l+1)$, by denoting $L = \lceil \log(n-m+1) \rceil - 1$ and $l = \lceil \log m \rceil$.*

Proof. Our goal is to evaluate a polynomial A of degree n in m evaluation points $\{y_1, ..., y_m\}$ homomorphically. To ease the description of the algorithm, we will assume that $m = 2^l$. We will consider the polynomials $P_{(\frac{i}{2^k})}$ presented in (55), that can also be defined with the following sequence:

$$(P) = \begin{cases} P_{(\frac{i}{m})} & = Z - y_i; & i \in \{1, ..., m\} \\ P_{(\frac{i}{2^{k-1}})} & = P_{(\frac{2i-1}{2^k})} P_{(\frac{2i}{2^k})}; & k \in \{1, ..., l\},\ i \in \{1, ..., 2^k\} \end{cases} \tag{67}$$

If now the $\{y_i\}$ are encrypted in a FHE scheme, one can compute those polynomials homomorphically and obtain the following sequence $(\widetilde{P})$:

$$(\widetilde{P}) = \begin{cases} \widetilde{P_{(\frac{i}{m})}} & = Z -_F \widetilde{y_i}; & i \in \{1, ..., m\} \\ \widetilde{P_{(\frac{i}{2^{k-1}})}} & = \widetilde{P_{(\frac{2i-1}{2^k})}} \times_F \widetilde{P_{(\frac{2i}{2^k})}}; & k \in \{1, ..., l\},\ i \in \{1, ..., 2^k\} \end{cases} \tag{68}$$

computing homomorphically all the $\left\{\widetilde{P_{(\frac{i}{2^k})}}\right\}$, for $k \in \{0, ..., l\}$ and $i \in \{1, ..., 2^k\}$ from the $\{\widetilde{y_i}\}_{i \in [m]}$ requires less than $\frac{1}{2} l \mathcal{M}_F(m) + \widetilde{O}(m)$ arithmetic operations with a depth l.

Through an adaptation of the Newton iterations, we can compute the tree of the $\left\{\overleftarrow{P_{(\frac{i}{2^{l-k}})}}^{-1} \mod Z^{2^k}\right\}$, for $k \in \{0, ..., l\}$, as following:

$$(V) = \begin{cases} V_0^{(i)} & = 1; & & i \in \{1, ..., m\} \\ V_1^{(i)} & = 1 + (y_{2i-1} + y_{2i})Z; & & i \in \{1, ..., 2^{l-1}\} \\ V_{k+1}^{(i)} & = K_0 - Z^{2^k} \left[[K_0]_0^{2^k-1} (K_1 + K_2) \right]_0^{2^k-1} & \mod Z^{2^{k+1}}; & i \in \{1, ..., 2^{l-(k+1)}\} \end{cases} \tag{69}$$

where, for the computation of $V_{k+1}^{(i)}$, we have

$$K_0 = V_k^{(2i-1)} V_k^{(2i)}, \tag{70}$$

$$K_1 = \left[\overleftarrow{P_{(\frac{2i-1}{2^{l-k}})}} V_k^{(2i-1)} \right]_{2^k}^{2^{k+1}-1}, \tag{71}$$

$$K_2 = \left[\overleftarrow{P_{(\frac{2i}{2^{l-k}})}} V_k^{(2i)} \right]_{2^k}^{2^{k+1}-1}. \tag{72}$$

We remark that or all $k \in \{0, ..., l\}$ and $i \in \{1, ..., 2^{l-k}\}$,

$$V_k^{(i)} = \overleftarrow{P_{\left(\frac{i}{2^{l-k}}\right)}}^{-1} \mod Z^{2^k} \tag{73}$$

Indeed, this is true for $k = 0$. For $k = 1$, we have

$$\overleftarrow{P_{\left(\frac{i}{2^{l-1}}\right)}} = \overleftarrow{(Z - y_{2i-1})(Z - y_{2i})} = 1 - (y_{2i-1} + y_{2i})Z + y_{2i-1}y_{2i}Z^2 \tag{74}$$

so the property is true for $k = 1$. Now, for $k + 1 > 1$ and $i \in \{1, ..., 2^{l-(k+1)}\}$,

$$\begin{aligned}
\overleftarrow{P_{\left(\frac{i}{2^{l-k+1}}\right)}} V_{k+1}^{(i)} =& \overleftarrow{P_{\left(\frac{2i-1}{2^{l-k}}\right)}} V_k^{(2i-1)} \overleftarrow{P_{\left(\frac{2i}{2^{l-k}}\right)}} V_k^{(2i)} - X^{2^k} \overleftarrow{P_{\left(\frac{i}{2^{l-k+1}}\right)}} \left[[K_0]_0^{2^k-1} (K_1 + K_2) \right]_0^{2^k-1} \mod Z^{2^{k+1}} \\
=& (1 + X^{2^k} K_1)(1 + X^{2^k} K_2) \\
& - X^{2^k} (K_1 + K_2) \left(\left[\overleftarrow{P_{\left(\frac{2i-1}{2^{l-k}}\right)}} V_k^{(2i-1)} \right]_0^{2^k-1} \left[\overleftarrow{P_{\left(\frac{2i}{2^{l-k}}\right)}} V_k^{(2i)} \right]_0^{2^k-1} \right) \mod Z^{2^{k+1}} \\
=& 1 \mod Z^{2^{k+1}}
\end{aligned}$$

Having the $\{y_i\}$ encrypted allow one to compute homomorphically the sequence (V) after the computation of the $\left\{\widetilde{P_{\left(\frac{i}{2^k}\right)}}\right\}$:

$$(\widetilde{V}) = \begin{cases}
\widetilde{V_0^{(i)}} & = \widetilde{1}; & & i \in \{1, ..., m\} \\
\widetilde{V_1^{(i)}} & = \widetilde{1} +_F (\widetilde{y_{2i-1}} + \widetilde{y_{2i}})Z; & & i \in \{1, ..., 2^{l-1}\} \\
\widetilde{V_{k+1}^{(i)}} & = \widetilde{K_0} - Z^{2^k} \left[\left[\widetilde{K_0} \right]_0^{2^k-1} \times_F (\widetilde{K_1} +_F \widetilde{K_2}) \right]_0^{2^k-1} & \mod Z^{2^{k+1}}; & i \in \{1, ..., 2^{l-(k+1)}\}
\end{cases} \tag{75}$$

where, for the computation of $\widetilde{V_{k+1}^{(i)}}$, we have

$$\widetilde{K_0} = \widetilde{V_k^{(2i-1)}} \widetilde{V_k^{(2i)}}, \tag{76}$$

$$\widetilde{K_1} = \left[\widetilde{\overleftarrow{P_{\left(\frac{2i-1}{2^{l-k}}\right)}}} \times_F \widetilde{V_k^{(2i-1)}} \right]_{2^k}^{2^{k+1}-1}, \tag{77}$$

$$\widetilde{K_2} = \left[\widetilde{\overleftarrow{P_{\left(\frac{2i}{2^{l-k}}\right)}}} \times_F \widetilde{V_k^{(2i)}} \right]_{2^k}^{2^{k+1}-1}. \tag{78}$$

computing homomorphically all the $\widetilde{V_k^{(i)}}$, for $k \in \{0, ..., l\}$ and $i \in \{1, ..., 2^{l-k}\}$ requires less than $2(l - 2)\mathcal{M}_F(m) + \widetilde{O}(m)$ arithmetic operations. Also, this algorithm has a depth $2(l - 1)$. In particular, we obtain homomorphically $\overleftarrow{P_{\left(\frac{1}{1}\right)}}^{-1} \mod Z^m$ as it is encrypted in $\widetilde{V_l^{(1)}}$. With another sequence of newton iteration, we want to obtain homomorphically $\overleftarrow{P_{\left(\frac{1}{1}\right)}}^{-1} \mod Z^{n-m+1}$ in order to perform the euclidean division of A by $P_{\left(\frac{1}{1}\right)} = \prod_{i=1}^{m} (Z - y_i)$. In fact, we are computing the sequence $(\widetilde{U})$ from (61):

$$(\widetilde{U}) = \begin{cases}
\widetilde{U_l} = \widetilde{V_l^{(1)}} \\
\widetilde{U_{k+1}} = \widetilde{U_k} \times_F \left(\widetilde{1} -_F \left[\widetilde{\overleftarrow{P_{\left(\frac{1}{1}\right)}}} \times_F \widetilde{U_k} \right]_{2^k}^{2^{k+1}-1} Z^{2^k} \right) & \mod Z^{2^{k+1}}
\end{cases} \tag{79}$$

As explained in Section 5.2.2, by denoting $L = \lceil \log(n - m + 1) \rceil - 1$ it requires less than $2\mathcal{M}_F(n - m) + O(n)$ arithmetic operations to obtain homomorphically $\widetilde{U_L}$, with a depth $2(L - l)$. With less than $2\mathcal{M}_F(n - m) + O(n)$ arithmetic operations and a depth 3, we compute homomorphically the remainder of A divided by

$P_{\left(\frac{1}{1}\right)}$, that we will name $\widetilde{R_l}$. We are now applying a multi-point evaluation algorithm different from the one presented in Section 5.2.1, which consists in successive euclidean remainders in the $P_{\left(\frac{i}{2^k}\right)}$. The previously computed $\widetilde{P_{\left(\frac{i}{2^k}\right)}}$ and $\widetilde{V_k^{(i)}}$ will help us to do this algorithm homomorphically through the following sequence:

$$(\widetilde{R}) = \begin{cases} \widetilde{R_l^{(1)}} & = \widetilde{R_l}; \\ \widetilde{R_k^{(2i-1)}} & = \widetilde{R_{k+1}^{(i)}} -_F \widetilde{P_{\left(\frac{2i-1}{2^{l-k}}\right)}} \left[\widetilde{V_k^{(2i-1)}} \overleftarrow{\widetilde{R_{k+1}^{(i)}}} \right]_0^{\overleftarrow{2^k-1}} & \mod Z^{2^k}; \quad k \in \{0,...,l-1\},\ i \in \{1,...,2^{l-k}\} \\ \widetilde{R_k^{(2i)}} & = \widetilde{R_{k+1}^{(i)}} -_F \widetilde{P_{\left(\frac{2i}{2^{l-k}}\right)}} \left[\widetilde{V_k^{(2i)}} \overleftarrow{\widetilde{R_{k+1}^{(i)}}} \right]_0^{2^k-1} & \mod Z^{2^k}; \quad k \in \{0,...,l-1\},\ i \in \{1,...,2^{l-k}\} \end{cases} \tag{80}$$

The sequence $(\widetilde{R})$ satisfies the following correctness, assuming sk_F is the decryption key, $\forall k \in \{0,...,l-1\}, \forall i \in \{1,...,2^{l-k}\}$:

$$\mathbf{F.D}_{sk_F}\left(\widetilde{R_{k+1}^{(i)}}\right) \mod P_{\left(\frac{2i-1}{2^{l-k}}\right)} = \mathbf{F.D}_{sk_F}\left(\widetilde{R_k^{(2i-1)}}\right), \tag{81}$$

$$\mathbf{F.D}_{sk_F}\left(\widetilde{R_{k+1}^{(i)}}\right) \mod P_{\left(\frac{2i}{2^{l-k}}\right)} = \mathbf{F.D}_{sk_F}\left(\widetilde{R_k^{(2i)}}\right). \tag{82}$$

In particular, we have :

$$\mathbf{F.D}_{sk_F}\left(\widetilde{R_0^{(i)}}\right) = A(y_i) \tag{83}$$

Finally, computing homomorphically all the $\left\{\widetilde{R_0^{(i)}}\right\}$ for $i \in \{1,...,m\}$, given $\widetilde{R_l}$, all the $\widetilde{P_{\left(\frac{i}{2^k}\right)}}$ and all the $\widetilde{V_k^{(i)}}$ requires less than $2l\mathcal{M}_F(m) + \widetilde{O}(m)$ arithmetic operations. Also, this algorithm has a depth $2l$.

In total, for $n > m$, less than $4\mathcal{M}_F(n-m) + O(n)$ arithmetic operations and a depth $2(L+l+1) = 2(\lceil\log(n-m+1)\rceil + \lceil\log m\rceil)$ are required. □

Remark 7. Similarly to Rem. 6, we can reduce de total depth of the algorithm with some extra information, under the hypothesis $\lceil\log m\rceil < 2(\lceil\log(n-m+1)\rceil - \lceil\log m\rceil) + 1$; this condition implies that the sequence $(\widetilde{U})$ plus the three homomorphic products to compute the remainder generate more depth than the sequence $(\widetilde{P})$. If one starts the sequence $(\widetilde{U})$ with a fresh encryption $\widetilde{U_l}$ of $\overleftarrow{\widetilde{P_{\frac{1}{1}}}^{-1}} \mod Z^{2^{\lceil\log m\rceil}}$, then the final depth will be reduced by $2\lceil\log m\rceil - 1$, leading to a total depth of $2\lceil\log(n-m+1)\rceil + 1$ (the arithmetic cost remains unchanged).

5.3.2 Protocol with F.MEv

Formally, our protocol is built with the algorithms **Setup**, **Encode**, **Reduce**, **Map** and **Union** respectively presented in Algs. 1, 4, 5, 9 and 10. A more visual version is presented in Protocol 3.

Theorem 3. *The protocol built with the algorithms **Setup**, **Encode**, **Reduce**, **Map** and **Union**, respectively presented in Algs. 1, 4, 5, 9 and 10, is a secure unbalanced private set union scheme (UPSU) under the honest-but-curious adversary model. For the receiver owning a set $\boldsymbol{X}$ of n plaintexts, and the sender owning a set $\boldsymbol{Y}$ of m plaintexts, with the assumption that $n > m$, it computes the set union with the asymptotic complexity bounds presented in Table 5.*

Algorithm 9 Encode$(\mathbf{Y}, keys_{\mathcal{S}})$

Input: A set of plaintexts $\mathbf{Y} = \{y_i\}_{i\in[m]} \subset \mathbb{M}$ and $keys_{\mathcal{S}} = \{(pk_F, sk_F), pk_L\}$.
Output: A set of ciphertexts $\{\widetilde{y_i}\}_{i\in[m]} \subset \mathbb{E}_F$ and an encrypted polynomial $\widetilde{U_l} \in \mathbb{E}_F[Z]$, such that, for all $i \in [m]$, $\mathbf{F.D}_{sk_F}(\widetilde{y_i}) = y_i$, and $\mathbf{F.D}_{sk_F}(\widetilde{U_l}) = \overleftarrow{\prod_{i\in[m]} (Z - y_i)}^{-1} \mod Z^{2^{\lceil \log m \rceil}}$.

1: $\mathcal{S}$: compute $P_{\mathcal{S}} \leftarrow \prod_{y\in\mathbf{Y}} (Z - y)$;
2: $\mathcal{S}$: compute $U_l \leftarrow \overleftarrow{P_{\mathcal{S}}}^{-1} \mod Z^{2^{\lceil \log m \rceil}}$;
3: $\mathcal{S}$: compute $\{\widetilde{y_i}\}_{i\in[m]} \leftarrow \mathbf{F.E}_{pk_F}(\mathbf{Y})$ and $\widetilde{U_l} \leftarrow \mathbf{F.E}_{pk_F}(P_{\mathcal{S}})$, and send $\widetilde{P_{\mathcal{S}}}, \widetilde{U_l}$ to $\mathcal{R}$;
4: $\mathcal{R}$: **return** $\{\{\widetilde{y_i}\}_{i\in[m]}, \widetilde{U_l}\}$;

Algorithm 10 Reduce$(\mathbf{X}, \{\{\widetilde{y_i}\}_{i\in[m]}, \widetilde{U_l}\}, keys_{\mathcal{R}})$

Input: A set of plaintext $\mathbf{X} = \{x_j\}_{j\in[n]} \subset \mathbb{M}$, a set of ciphertexts $\{\{\widetilde{y_i}\}_{i\in[m]}, \widetilde{U_l}\} \subset \mathbb{E}_F$ and $keys_{\mathcal{R}} = \{(pk_L, sk_L), pk_F\}$.
Output: Two sets of ciphertexts $\{\widetilde{h_i}\}_{i\in[m]} \subset \mathbb{E}_F$ and $\{\widehat{k_i}\}_{i\in[m]} \subset \mathbb{E}_L$, such that for $i \in [m]$, $\mathbf{F.D}_{sk_F}(\widetilde{h_i}) - \mathbf{L.D}_{sk_L}(\widehat{k_i}) = \prod_{j\in[n]} (\mathbf{F.D}_{sk_F}(\widetilde{y_i}) - x_j)$.

1: $\mathcal{R}$: compute $P_{\mathcal{R}} \leftarrow \prod_{j\in[n]} (Z - x_j)$;
2: **for all** $i \in [m]$ **do**
3: $\quad\mathcal{R}$: randomly select $k_i \xleftarrow{\$} \mathbb{M}$;
4: $\quad\mathcal{R}$: compute $\widehat{k_i} \leftarrow \mathbf{L.E}_{pk_L}(k_i)$;
5: $\quad\mathcal{R}$: compute $\widetilde{k_i} \leftarrow \mathbf{F.E}_{pk_F}(k_i)$;
6: **end for**
7: $\mathcal{R}$: compute $\{\widetilde{e_i}\}_{i\in[m]} \leftarrow \mathbf{F.MEv}(P_{\mathcal{R}}, \{\widetilde{y_i}\}_{i\in[m]})$;
8: **for all** $i \in [m]$ **do**
9: $\quad\mathcal{R}$: compute $\widetilde{h_i} \leftarrow \widetilde{k_i} +_F \widetilde{e_i}$;
10: **end for**
11: $\mathcal{R}$: send $\{\widetilde{h_i}\}_{i\in[m]}$ and $\{\widehat{k_i}\}_{i\in[m]}$ to $\mathcal{S}$;
12: $\mathcal{S}$: **return** $\{\{\widetilde{h_i}\}_{i\in[m]}, \{\widehat{k_i}\}_{i\in[m]}\}$;

Table 5: Cost analysis of Protocol 3 for $n > m$

Algorithm	Ar. Cost for $\mathcal{R}$	Ar. Cost for $\mathcal{S}$	Comm. Vol.	Depth
Setup	$O(1)$	$O(1)$	$O(1)$	
Encode	$O(1)$	$\widetilde{O}(m)$	$O(m)$	
Reduce	$4\mathcal{M}_F(n) + O(n)$	$O(1)$	$O(m)$	$2\log n + 1$
Map	$O(1)$	$O(m)$	$O(m)$	
Union	$O(m)$	$O(1)$	$O(1)$	
Total	$4\mathcal{M}_F(n) + O(n)$	$\widetilde{O}(m)$	$O(m)$	$2\log n + 1$

Proof. The complete simulation proof is presented in Appendix A.3. The asymptotic presented in Table 5 are implied by Prop. 7 (together with the analysis of Rem. 7). The correctness of the protocol is basically the same than the one of Protocol 1. The idea is exactly the same, the receiver evaluates homomorphically the polynomial $P_{\mathcal{R}}$, whose roots are its elements, in all the encrypted sender's elements, and in **Union**, it can compute only and exactly the evaluations points that were not roots. The one difference is the algorithm used to perform homomorphically the multi-point evaluation, so the correctness of **F.MEv** implies the correctness of the protocol. □

Protocol 3: Optimal communication and arithmetic cost with compatibility LHE-FHE protocol

	$\mathcal{R}$		$\mathcal{S}$	
	$\mathbf{X} = \{x_1, ..., x_n\}$		$\mathbf{Y} = \{y_1, ..., y_m\}$	
Setup	$\{pk_F, sk_L, pk_L\}$	$\longleftrightarrow$	$\{pk_L, sk_F, pk_F\}$	**Setup**
	$P_{\mathcal{R}} \leftarrow \prod(Z - x_i)$		$P_{\mathcal{S}} \leftarrow \prod(Z - y_i)$	
			$U_l \leftarrow \overleftarrow{P_{\mathcal{S}}}^{-1} \mod Z^{2^{\lceil \log m \rceil}}$	
			$\{\widetilde{y_i}\}_{i\in[m]} \leftarrow \mathbf{F.E}_{pk_F}(\{y_i\}_{i\in[m]})$	
Reduce	$\{k_i\}_{i\in[m]} \xleftarrow{\$} \mathbb{M}_F$	$\xleftarrow{\{\widetilde{y_i}\}_{i\in[m]},\ \widetilde{U_l}}$	$\widetilde{U_l} \leftarrow \mathbf{F.E}_{pk_F}(U_l)$	**Encode**
	$\{\widehat{k_i}\}_{i\in[m]} \leftarrow \{\mathbf{L.E}_{pk_L}(k_i)\}_{i\in[m]}$			
	$\{\widetilde{k_i}\}_{i\in[m]} \leftarrow \{\mathbf{F.E}_{pk_F}(k_i)\}_{i\in[m]}$			
	$\{\widetilde{e_i}\}_{i\in[m]} \leftarrow \mathbf{F.MEv}(P_{\mathcal{R}}, \{\widetilde{y_i}\}_{i\in[m]})$			
	$\{\widetilde{h_i}\}_{i\in[m]} \leftarrow \{\widetilde{k_i} +_F \widetilde{e_i}\}_{i\in[m]}$	$\xrightarrow{\{\widetilde{h_i}\}_{i\in[m]},\ \{\widehat{k_i}\}_{i\in[m]}}$	$\{h_i\}_{i\in[m]} \leftarrow \{\mathbf{F.D}_{sk_F}(\widetilde{h_i})\}_{i\in[m]}$	**Map**
			$\pi \xleftarrow{\$} \mathfrak{S}_m$	
			$\{\widehat{e_{\pi(i)}}\}_{i\in[m]} \leftarrow \{\widehat{h_i} -_L \widehat{k_i}\}_{i\in[m]}$	
Union	$\{e_j\}_{j\in[m]} \leftarrow \{\mathbf{L.D}_{sk_L}(\widehat{e_j})\}_{j\in[m]}$	$\xleftarrow{\{(\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}})\}_{i\in[m]}}$	$\{\widehat{\eta_{\pi(i)}}\}_{i\in[m]} \leftarrow \{y_i \ltimes_L \widehat{e_{\pi(i)}}\}_{i\in[m]}$	
	$\begin{cases} e_j = 0 & \Rightarrow \quad \perp \\ e_j \neq 0 & \Rightarrow \quad \mathbf{X} \leftarrow \mathbf{X} + \{\mathbf{L.D}_{sk_L}(\widehat{\eta_j})\, e_j^{-1}\} \end{cases}$			
	Return X			

References

[Bloom(1970)] Burton H. Bloom. 1970. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM* 13, 7 (1970), 422–426. https://doi.org/10.1145/362686.362692

[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.

[Chen et al.(2018)] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. 2018. Labeled PSI from Fully Homomorphic Encryption with Malicious Security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM, 1223–1237. https://doi.org/10.1145/3243734.3243836

[Chou and Orlandi(2015)] Tung Chou and Claudio Orlandi. 2015. The Simplest Protocol for Oblivious Transfer. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings (Lecture Notes in Computer Science, Vol. 9230)*, Kristin E. Lauter and Francisco Rodríguez-Henríquez (Eds.). Springer, 40–58. https://doi.org/10.1007/978-3-319-22174-8_3

[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

[Fan and Vercauteren(2012)] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptol. ePrint Arch.* (2012), 144. http://eprint.iacr.org/2012/144

[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.(2021a)] Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. 2021a. 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

[Garimella et al.(2021b)] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. 2021b. Oblivious Key-Value Stores and Amplification for Private Set Intersection. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12826)*, Tal Malkin and Chris Peikert (Eds.). Springer, 395–425. https://doi.org/10.1007/978-3-030-84245-1_14

[Goldreich(2001)] Oded Goldreich. 2001. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press. https://doi.org/10.1017/CBO9780511546891

[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

[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

[Naor and Pinkas(2001)] Moni Naor and Benny Pinkas. 2001. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, S. Rao Kosaraju (Ed.). ACM/SIAM, 448–457. http://dl.acm.org/citation.cfm?id=365411.365502

[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

[von zur Gathen and Gerhard(2013)] Joachim von zur Gathen and Jürgen Gerhard. 2013. *Modern Computer Algebra (3. ed.)*. Cambridge University Press.

[Yi et al.(2014)] Xun Yi, Russell Paulet, and Elisa Bertino. 2014. *Homomorphic Encryption and Applications*. Springer. https://doi.org/10.1007/978-3-319-12229-8

[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 both IND-CPA secure. The oblivious transfer is assumed to be a black box secured in honest-but-curious adversary model, so we consider that the view generated by it for each party can be reduced to their input and output. 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 $f : \mathcal{P}(\mathbb{M}_F) \times \mathcal{P}(\mathbb{M}_F) \to (\mathcal{P}(\mathbb{M}_F) \times \mathbb{N}) \times \mathcal{P}(\mathbb{M}_F)$ where $\mathcal{P}$ denotes the power set and $\mathbb{M}_F$ is the FHE plaintext space. 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 $\mathbf{F.E}_{pk}(R)$. In the following, we consider that the **Setup** algorithm has been done as it is basically an exchange of two secret keys sk_L and sk_F, respectively for LHE and FHE scheme, and a hand-check on the map Ψ presented in Section 4.5. The views and the outputs of each parties are:

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

$$C_1 = \left\{ pk_L, sk_L, pk_F, \{k_i\}_{i\in[m]}, \widetilde{\mathbf{e}}, \{e_{\pi(i)}\}_{i\in[m]}, \mathcal{D} \right\}, \tag{84}$$

  where $\widetilde{\mathbf{e}} \leftarrow \mathbf{F.BSMEv}(P_{\mathcal{R}}, \widetilde{\mathbf{y}}, \sqrt{n})$, for all $j \in [m]$, $e_j \leftarrow \mathbf{L.D}_{sk_L}(\widehat{e_j})$, and $\mathcal{D}$ is a data set depending on the three different situations presented in Section 4.5. In **Situation 1**,

$$\mathcal{D} = \left\{ e_{\pi(i)}, \mathbf{L.D}_{sk_L}(\widehat{\eta_{\pi(i)}}) \right\}_{i\in[m]}, \tag{85}$$

  in **Situation 2**,

$$\mathcal{D} = \left\{ e_{\pi(i)}, \mathbf{L.D}_{sk_L}(\widehat{\eta_{\pi(i)}}), \mathbf{L.D}_{sk_L}(\widehat{\nu_{\pi(i)}}) \right\}_{i\in[m]}, \tag{86}$$

  and in **Situation 3**,

$$\mathcal{D} = \left\{ \mu_{\pi(i)} \right\}_{i\in[m]}. \tag{87}$$

  The content of the first message is:

$$M_1 = \{\widetilde{\mathbf{y}}\}, \tag{88}$$

  while the content of the second message depends on the situations of Section 4.5. In **Situation 1**,

$$M_3 = \left\{ \{\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}}\}_{i\in[m]} \right\}, \tag{89}$$

  for $\widehat{e_{\pi(i)}} \leftarrow \mathbf{L.E}_{pk_L}(P_{\mathcal{R}}(y_{\pi(i)}))$ and $\widehat{\eta_{\pi(i)}} \leftarrow y_{\pi(i)} \ltimes_L \widehat{e_{\pi(i)}}$. In **Situation 2**,

$$M_3 = \left\{ \{\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}}, \widehat{\nu_{\pi(i)}}\}_{i\in[m]} \right\}, \tag{90}$$

for $\widehat{e_{\pi(i)}} \leftarrow \mathbf{L.E}_{pk_L}(\Psi(P_{\mathcal{R}}(y_{\pi(i)}) + k_{\pi(i)})) -_L \mathbf{L.E}_{pk_L}(\Psi(k_{\pi(i)}))$, $\widehat{\eta_{\pi(i)}} \leftarrow \Psi(y_{\pi(i)}) \ltimes_L \widehat{e_{\pi(i)}}$ and $\widehat{\nu_{\pi(i)}} \leftarrow \Psi(y_{\pi(i)}) \ltimes_L (\mathbf{L.E}_{pk_L}(0) -_L \widehat{e_{\pi(i)}})$. In **Situation 3**,

$$M_3 = \left\{\{\widehat{e_{\pi(i)}}\}_{i\in[m]}\right\}, \tag{91}$$

for $\widehat{e_{\pi(i)}} \leftarrow \mathbf{L.E}_{pk_L}(\Psi(P_{\mathcal{R}}(y_{\pi(i)}) + k_{\pi(i)})) -_L \mathbf{L.E}_{pk_L}(\Psi(k_{\pi(i)}))$.

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

$$C_2 = \left\{pk_F, sk_F, pk_L, \begin{pmatrix} h_1 & \cdots & h_m \end{pmatrix}, \{\widehat{e_i}\}_{i\in[m]}, \pi\right\},$$
$$M_2 = \left\{\widetilde{\mathbf{h}}, \{\widehat{k_i}\}_{i\in[m]}\right\},$$

  where $\begin{pmatrix} h_1 & \cdots & h_m \end{pmatrix} \leftarrow \mathbf{F.D}_{sk_F}(\widetilde{\mathbf{h}})$.

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

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

$$S_2(\mathbf{Y}, \emptyset) = \Bigg(\mathbf{Y}, \left\{pk_F, sk_F, pk_L, \begin{pmatrix} r_1 & \cdots & r_m \end{pmatrix}, \{\mathbf{L.E}_{pk_L}(\Psi(r_i)) -_L \mathbf{L.E}_{pk_L}(\Psi(r'_i))\}_{i\in[m]}, \pi'\right\},$$
$$\left\{\mathbf{F.Batch}(\begin{pmatrix} \mathbf{low}(\mathbf{F.E}_{pk_F}(r_1)) & \cdots & \mathbf{low}(\mathbf{F.E}_{pk_F}(r_m)) \end{pmatrix}), \{\mathbf{L.E}_{pk_L}(\Psi(r'_i))\}_{i\in[m]}\right\}\Bigg),$$

where $m = |\mathbf{Y}|$, pk_F, sk_F, pk_L are obtained from the **Setup** algorithm, $r_i, r'_i \xleftarrow{\$} \mathbb{M}_F$ for all $i \in [m]$, and $\pi' \xleftarrow{\$} \mathfrak{S}_m$.
In the protocol, each entries of $\begin{pmatrix} h_1 & \cdots & h_m \end{pmatrix}$ is the result of a sum with a random plaintext $k_i \xleftarrow{\$} \mathbb{M}_F$, so in particular, each entries is indistinguishable from a random plaintext $r_i \xleftarrow{\$} \mathbb{M}_F$. It implies, as $\begin{pmatrix} h_1 & \cdots & h_m \end{pmatrix}$ is the decryption of a batched ciphertext $\widetilde{\mathbf{h}}$, that an encryption of $\begin{pmatrix} r_1 & \cdots & r_m \end{pmatrix}$ batched is indistinguishable from $\widetilde{\mathbf{h}}$; notice that $\widetilde{\mathbf{h}}$ is at the lowest depth in the protocol, so we have to use the algorithm **low** on the simulation ciphertexts. A random permutation π is indistinguishable from a random permutation π', and $\{\widehat{k_i}\}_{i\in[m]}$ is a tuple containing encryptions of m random plaintexts $k_i \xleftarrow{\$} \mathbb{M}_F$ with the key pk_L, so, as the LHE scheme is assumed IND-CPA, it is easily simulated with a tuple containing encryptions of m random plaintexts $r'_i \xleftarrow{\$} \mathbb{M}_F$ (using the one-to-one correspondence Ψ). It is clear that $\{\widehat{e_i}\}_{i\in[m]}$ is indistinguishable from $\{\mathbf{L.E}_{pk_L}(\Psi(r_i)) -_L \mathbf{L.E}_{pk_L}(\Psi(r'_i))\}_{i\in[m]}$ as it is constructed the same way with indistinguishable parts. Note that, in the protocol, $|\mathbf{X} \cap \mathbf{Y}|$ of the $\widehat{e_i}$ are encryptions of 0. However, as the LHE scheme is assumed IND-CPA, those encryptions of zeros are indistinguishable from encryptions of anything else. Overall, we have shown that for every subsets $\mathbf{X}, \mathbf{Y} \subset \mathbb{M}_F$:

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

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

$$S_1(\mathbf{X}, (\mathbf{X} \cup \mathbf{Y}, |\mathbf{Y}|)) = \Bigg(\mathbf{X}, \left\{pk_L, sk_L, pk_F, \{r_i\}_{i\in[m]}, \mathbf{F.BSMEv}(P_{\mathcal{R}}, \widetilde{\mathbf{z}}, \sqrt{n}), \{\zeta_{\pi'(i)}\}_{i\in[m]}, \mathcal{D}'\right\},$$
$$\{\widetilde{\mathbf{z}}\}, \{\mathcal{T}\}\Bigg).$$

Let $\{z_i\}_{i\in[m]} \subset \mathbb{M}_F$ be a set of $m = |\mathbf{Y}|$ distinct plaintexts, such that exactly $|\mathbf{Y}| - (|\mathbf{X} \cup \mathbf{Y}| - |\mathbf{X}|) = |\mathbf{X} \cap \mathbf{Y}|$ of the z_i (uniformly distributed) are randomly picked in $\mathbf{X}$ and the others are the plaintexts in $\mathbf{X} \cup \mathbf{Y} \setminus \mathbf{X}$.

pk_L, sk_L, pk_F are obtained from the **Setup** algorithm, $\widetilde{\mathbf{z}} \leftarrow \mathbf{F.Batch}(\begin{pmatrix}\mathbf{F.E}_{pk_F}(z_1) & \cdots & \mathbf{F.E}_{pk_F}(z_m)\end{pmatrix})$ is a batched encryption of the z_i, $P_{\mathcal{R}}$ is the plaintext polynomial whose roots are the elements of $\mathbf{X}$, $\pi' \xleftarrow{\$} \mathfrak{S}_m$ is a random permutation of m elements, $\{r_i\}_{i\in[m]}$ is a set of m random plaintexts in $\mathbb{M}_F$, $\zeta_j = \Psi(P_{\mathcal{R}}(z_j) + r_j) - \Psi(r_j)$ for all $j \in [m]$, and $\mathcal{D}'$ and $\mathcal{T}$ depend on the situations presented in Section 4.5 as following. In **Situation 1**,

$$\mathcal{D}' = \{\zeta_{\pi'(i)}, z_{\pi'(i)}\zeta_{\pi'(i)}\}_{i\in[m]}, \tag{92}$$
$$\mathcal{T} = \{\mathbf{L.E}_{pk_L}(\zeta_{\pi'(i)}), \mathbf{L.E}_{pk_L}(z_{\pi'(i)}\zeta_{\pi'(i)})\}_{i\in[m]}. \tag{93}$$

In **Situation 2**,

$$\mathcal{D}' = \{\zeta_{\pi'(i)}, \Psi(z_{\pi'(i)})\zeta_{\pi'(i)}, \Psi(z_{\pi'(i)})(N - \zeta_{\pi'(i)})\}_{i\in[m]}, \tag{94}$$
$$\mathcal{T} = \{\mathbf{L.E}_{pk_L}(\zeta_{\pi'(i)}), \mathbf{L.E}_{pk_L}(\Psi(z_{\pi'(i)})\zeta_{\pi'(i)}), \mathbf{L.E}_{pk_L}(\Psi(z_{\pi'(i)})(N - \zeta_{\pi'(i)}))\}_{i\in[m]}. \tag{95}$$

In **Situation 3**,

$$\mathcal{D}' = \{\mu'_{\pi'(i)}\}_{i\in[m]}, \tag{96}$$
$$\mathcal{T} = \{\mathbf{L.E}_{pk_L}(\zeta_{\pi'(i)})\}_{i\in[m]}, \tag{97}$$

where, for all $i \in [m]$, $\mu'_{\pi'(i)} = \perp$ if $z_i \in \mathbf{X}$, and if not, $\mu'_{\pi'(i)} = z_i$.
In the protocol, $\{k_i\}_{i\in[m]}$ is a set of m random plaintexts in $\mathbb{M}_F$, so it is indistinguishable from $\{r_i\}_{i\in[m]} \subset \mathbb{M}_F$. The goal is to prove that $\{z_i\}_{i\in[m]}$ is indistinguishable from $\mathbf{Y} = \{y_i\}_{i\in[m]}$. First, $\mathbf{X} \cup \mathbf{Y} \setminus \mathbf{X}$ is a subset of both sets, by construction. Due to the IND-CPA security of the FHE scheme, $\widetilde{\mathbf{y}}$ is indistinguishable from $\widetilde{\mathbf{z}}$ and even by knowing that a plaintext $w \in \mathbf{X} \cup \mathbf{Y} \setminus \mathbf{X}$ is encrypted in both batched ciphertext, it is not possible to find its position in any batch; in fact, even if its position in one batched ciphertext is known, it is not possible to find its position in the other. In the **Union** algorithm, the receiver has a data set $\mathcal{D}$ containing m indexed tuples or plaintext elements. The indexes for which it can compute an element $y \in \mathbf{X} \cup \mathbf{Y} \setminus \mathbf{X}$ are $\{j \in [m] | P_{\mathcal{R}}(y_{\pi^{-1}(j)}) \neq 0\}$ with π an unknown permutation, so indistinguishable from another random permutation π': so in particular, $\{i \in [m] | y_i \in \mathbf{X} \cup \mathbf{Y} \setminus \mathbf{X}\}$ is indistinguishable from a random choice of $|\mathbf{X} \cup \mathbf{Y} \setminus \mathbf{X}|$ integers among $[m]$, so indistinguishable from $\{i \in [m] | z_i \in \mathbf{X} \cup \mathbf{Y} \setminus \mathbf{X}\}$. This analysis implies that $\{e_i | e_i \neq 0\}$ is indistinguishable from $\{\zeta_i | \zeta_i \neq 0\}$. By extension, $\{e_{\pi(i)}\}_{i\in[m]}$ is indistinguishable from $\{\zeta_{\pi'(i)}\}_{i\in[m]}$ as it only consists in the adjunction of 0 in the remaining indexes (by construction, $\zeta_j = 0 \Leftrightarrow z_j \in \mathbf{X}$). Finally, we have shown that $\{z_i\}_{i\in[m]}$ is indistinguishable from $\{y_i\}_{i\in[m]}$ and it implies all the remaining indistinguishabilities. Overall, we obtain for every subsets $\mathbf{X}, \mathbf{Y} \subset \mathbb{M}_F$:

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

A.2 Security Proof for Protocol 2

We assume that both the FHE and LHE schemes used are both IND-CPA 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 and $\mathbb{M}$ is the FHE and the LHE plaintext space. 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 $\mathbf{F.E}_{pk}(R)$. In the following, we consider that the **Setup** algorithm has been done as it is basically an exchange of two secret keys sk_L and sk_F, respectively for LHE and FHE scheme. The views and the outputs of each parties are:

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

$$C_1 = \left\{pk_L, sk_L, pk_F, M, \widetilde{R}, \{(e_{\pi(i)}, \eta_{\pi(i)})\}_{i\in[m]}\right\}, \tag{98}$$

where $\widetilde{R} \leftarrow P_{\mathcal{R}} \mathbf{mod}_F \left(\widetilde{P_{\mathcal{S}}}\right)$ and for all $j \in [m]$, $(e_j, \eta_j) \leftarrow \mathbf{L.D}_{sk_L}(\widehat{e_j}, \widehat{\eta_j})$. The content of the first message is:

$$M_1 = \left\{\widetilde{P_{\mathcal{S}}}, \widetilde{U_l}\right\}, \tag{99}$$

while the content of the second message is:

$$M_3 = \left\{\{\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}}\}_{i\in[m]}\right\}, \tag{100}$$

for $\widehat{e_{\pi(i)}} \leftarrow \mathbf{L.E}_{pk_L}(R(y_{\pi(i)}))$, where $R = (P_{\mathcal{R}} \mod P_{\mathcal{S}})$, and $\widehat{\eta_{\pi(i)}} \leftarrow y_{\pi(i)} \ltimes_L \widehat{e_{\pi(i)}}$.

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

$$C_2 = \left\{pk_F, sk_F, pk_L, H, \{\widehat{m_i}\}_{i\in[m]}, \{\widehat{e_i}\}_{i\in[m]}, \pi\right\},$$
$$M_2 = \left\{\widetilde{H}, \widehat{M}\right\},$$

  where $H \leftarrow \mathbf{F.D}_{sk_F}(\widetilde{H})$, $\{\widehat{m_i}\}_{i\in[m]} \leftarrow \mathbf{L.MEv}(\widehat{M}, \mathbf{Y})$ and for all $i \in [m]$, $\widehat{e_i} \leftarrow \mathbf{L.E}_{pk_L}(H(y_i)) -_L \widehat{m_i}$.

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

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

$$S_2(\mathbf{Y}, \emptyset) = \Big(\mathbf{Y}, \left\{pk_F, sk_F, pk_L, H', \{\mathbf{L.E}_{pk_L}(M'(y_i))\}_{i\in[m]}, \{\mathbf{L.E}_{pk_L}(H'(y_i)) -_L \mathbf{L.E}_{pk_L}(M'(y_i))\}_{i\in[m]}, \mathbf{Y}\right), \pi'\Big\},$$
$$\{\mathbf{low}(\mathbf{F.E}_{pk_F}(H')), \mathbf{L.E}_{pk_L}(M')\}\Big),$$

where pk_F, sk_F, pk_L are obtained from the **Setup** algorithm, $H', M' \xleftarrow{\$} \mathbb{M}[Z]$ are two random plaintext polynomials of degree $|\mathbf{Y}| - 1$, and $\pi' \xleftarrow{\$} \mathfrak{S}_m$.
In the protocol, H is the result of the remainder of $P_{\mathcal{R}}$ divided by $P_{\mathcal{S}}$, whose degree is at most $|\mathbf{Y}| - 1$, plus a random plaintext polynomial $M \xleftarrow{\$} \mathbb{M}[Z]$ of degree $|\mathbf{Y}| - 1$, so in particular, it is indistinguishable from a random plaintext polynomial $H' \xleftarrow{\$} \mathbb{M}[Z]$ of same degree. However, as the $\widetilde{H}$ received in the protocol is at the lowest depth, it is important to use **low** on the encryption of H' to simulate it. Obviously, $M' \xleftarrow{\$} \mathbb{M}[Z]$ of degree $|\mathbf{Y}| - 1$ is indistinguishable from M, so for all $i \in [m]$, $\mathbf{L.E}_{pk_L}(M'(y_i))$ is indistinguishable from $\widehat{m_i}$. By extension, for all $i \in [m]$, $\mathbf{L.E}_{pk_L}(H'(y_i)) -_L \mathbf{L.E}_{pk_L}(M'(y_i))$ simulates properly $\widehat{e_i}$. Note that, in the protocol, $|\mathbf{X} \cap \mathbf{Y}|$ of the $\widehat{e_i}$ are encryptions of 0. However, as the LHE scheme is assumed IND-CPA, those encryptions of zeros are indistinguishable from encryptions of anything else. Overall, we have shown that 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 $(\mathbf{X} \cup \mathbf{Y}, |\mathbf{Y}|)$ simulates $\mathbf{view}_1^{\Pi}(\mathbf{X}, \mathbf{Y})$ this way:

$$S_1(\mathbf{X}, (\mathbf{X} \cup \mathbf{Y}, |\mathbf{Y}|)) = \Big(\mathbf{X}, \left\{pk_L, sk_L, pk_F, M', \mathbf{F.E}_{pk_F}(R'), \{(R'(z_i), R'(z_i)z_i)\}_{i\in[m]}\right\},$$
$$\left\{\mathbf{F.E}(P_z), \mathbf{F.E}\left(\overleftarrow{P_z}^{-1} \mod Z^{2^{\lceil \log |\mathbf{Y}| \rceil}}\right)\right\},$$
$$\left\{\{(\mathbf{L.E}_{pk_L}(R'(z_{\pi(i)})), \mathbf{L.E}_{pk_L}(R'(z_{\pi'(i)})z_{\pi'(i)}),)\}_{i\in[m]}\right\}\Big).$$

Let $\{z_i\}_{i\in[m]} \subset \mathbb{M}$ be a set of $m = |\mathbf{Y}|$ distinct plaintexts, such that exactly $|\mathbf{Y}| - (|\mathbf{X}\cup\mathbf{Y}| - |\mathbf{X}|) = |\mathbf{X}\cap\mathbf{Y}|$ of the z_i (uniformly distributed) are randomly picked in $\mathbf{X}$ and the others are the plaintexts in $\mathbf{X}\cup\mathbf{Y}\setminus\mathbf{X}$. pk_L, sk_L, pk_F are obtained from the **Setup** algorithm, $M' \xleftarrow{\$} \mathbb{M}[Z]$ is a random plaintext polynomial of degree $m-1$, $P_z := \prod(Z - z_i)$, $R' := P_{\mathcal{R}} \mod P_z$ and $\pi' \xleftarrow{\$} \mathfrak{S}_m$ is a random permutation of m elements. With the same arguments than in Appendix A.1, the set $\{z_i\}_{i\in[m]} \subset \mathbb{M}$ is indistinguishable from $\mathbf{Y}$. So by doing everything the same way with the simulated input set $\{z_i\}_{i\in[m]}$, the entire view is indistinguishable. Overall, we obtain for every subsets $\mathbf{X}, \mathbf{Y} \subset \mathbb{M}_F$:

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

A.3 Security Proof for Protocol 3

We assume that both the FHE and LHE schemes used are both IND-CPA secure. The oblivious transfer is assumed to be a black box secured in honest-but-curious adversary model, so we consider that the view generated by it for each party can be reduced to their input and output. 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 $f : \mathcal{P}(\mathbb{M}_F) \times \mathcal{P}(\mathbb{M}_F) \to (\mathcal{P}(\mathbb{M}_F) \times \mathbb{N}) \times \mathcal{P}(\mathbb{M}_F)$ where $\mathcal{P}$ denotes the power set and $\mathbb{M}_F$ is the FHE plaintext space. 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 $\mathbf{F.E}_{pk}(R)$. In the following, we consider that the **Setup** algorithm has been done as it is basically an exchange of two secret keys sk_L and sk_F, respectively for LHE and FHE scheme, and a hand-check on the map Ψ presented in Section 4.5. The views and the outputs of each parties are:

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

$$C_1 = \left\{pk_L, sk_L, pk_F, \{k_i\}_{i\in[m]}, \{\widetilde{e_i}\}_{i\in[m]}, \{e_{\pi(i)}\}_{i\in[m]}, \mathcal{D}\right\}, \tag{101}$$

  where $\{\widetilde{e_i}\}_{i\in[m]} \leftarrow \mathbf{F.MEv}(P_{\mathcal{R}}, \{\widetilde{y_i}\}_{i\in[m]})$, for all $j \in [m]$, $e_j \leftarrow \mathbf{L.D}_{sk_L}(\widehat{e_j})$, and $\mathcal{D}$ is a data set depending on the three different situations presented in Section 4.5. In **Situation 1**,

$$\mathcal{D} = \left\{e_{\pi(i)}, \mathbf{L.D}_{sk_L}(\widehat{\eta_{\pi(i)}})\right\}_{i\in[m]}, \tag{102}$$

  in **Situation 2**,

$$\mathcal{D} = \left\{e_{\pi(i)}, \mathbf{L.D}_{sk_L}(\widehat{\eta_{\pi(i)}}), \mathbf{L.D}_{sk_L}(\widehat{\nu_{\pi(i)}})\right\}_{i\in[m]}, \tag{103}$$

  and in **Situation 3**,

$$\mathcal{D} = \left\{\mu_{\pi(i)}\right\}_{i\in[m]}. \tag{104}$$

  The content of the first message is:

$$M_1 = \left\{\{\widetilde{y_i}\}_{i\in[m]}, \widetilde{U_l}\right\}, \tag{105}$$

  while the content of the second message depends on the situations of Section 4.5. In **Situation 1**,

$$M_3 = \left\{\{\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}}\}_{i\in[m]}\right\}, \tag{106}$$

  for $\widehat{e_{\pi(i)}} \leftarrow \mathbf{L.E}_{pk_L}(P_{\mathcal{R}}(y_{\pi(i)}))$ and $\widehat{\eta_{\pi(i)}} \leftarrow y_{\pi(i)} \ltimes_L \widehat{e_{\pi(i)}}$. In **Situation 2**,

$$M_3 = \left\{\{\widehat{e_{\pi(i)}}, \widehat{\eta_{\pi(i)}}, \widehat{\nu_{\pi(i)}}\}_{i\in[m]}\right\}, \tag{107}$$

for $\widehat{e_{\pi(i)}} \leftarrow \mathbf{L.E}_{pk_L}(\Psi(P_{\mathcal{R}}(y_{\pi(i)}) + k_{\pi(i)})) -_L \mathbf{L.E}_{pk_L}(\Psi(k_{\pi(i)}))$, $\widehat{\eta_{\pi(i)}} \leftarrow \Psi(y_{\pi(i)}) \ltimes_L \widehat{e_{\pi(i)}}$ and $\widehat{\nu_{\pi(i)}} \leftarrow \Psi(y_{\pi(i)}) \ltimes_L (\mathbf{L.E}_{pk_L}(0) -_L \widehat{e_{\pi(i)}})$. In **Situation 3**,

$$M_3 = \left\{\{\widehat{e_{\pi(i)}}\}_{i\in[m]}\right\}, \tag{108}$$

for $\widehat{e_{\pi(i)}} \leftarrow \mathbf{L.E}_{pk_L}(\Psi(P_{\mathcal{R}}(y_{\pi(i)}) + k_{\pi(i)})) -_L \mathbf{L.E}_{pk_L}(\Psi(k_{\pi(i)}))$.

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

$$C_2 = \left\{pk_F, sk_F, pk_L, \{h_i\}_{i\in[m]}, \{\widehat{e_i}\}_{i\in[m]}, \pi\right\},$$
$$M_2 = \left\{\{\widetilde{h_i}\}_{i\in[m]}, \{\widehat{k_i}\}_{i\in[m]}\right\},$$

  where $\{h_i\}_{i\in[m]} \leftarrow \{\mathbf{F.D}_{sk_F}(\widetilde{h_i})\}_{i\in[m]}$.

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

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

$$S_2(\mathbf{Y}, \emptyset) = \Bigg(\mathbf{Y}, \left\{pk_F, sk_F, pk_L, \{r_i\}_{i\in[m]}, \{\mathbf{L.E}_{pk_L}(\Psi(r_i)) -_L \mathbf{L.E}_{pk_L}(\Psi(r'_i))\}_{i\in[m]}, \pi'\right\},$$
$$\left\{\{\mathbf{low}\left(\mathbf{F.E}_{pk_R}(r_i)\right)\}_{i\in[m]}, \{\mathbf{L.E}_{pk_L}(\Psi(r'_i))\}_{i\in[m]}\right\}\Bigg),$$

where $m = |\mathbf{Y}|$, pk_F, sk_F, pk_L are obtained from the **Setup** algorithm, $r_i, r'_i \xleftarrow{\$} \mathbb{M}_F$ for all $i \in [m]$, and $\pi' \xleftarrow{\$} \mathfrak{S}_m$.
In the protocol, for all $i \in [m]$, h_i is the result of a sum with a random plaintext $k_i \xleftarrow{\$} \mathbb{M}_F$, so in particular, each entries is indistinguishable from a random plaintext $r_i \xleftarrow{\$} \mathbb{M}_F$. It implies, as h_i is the decryption of a ciphertext $\widetilde{h_i}$, that an encryption of r_i is indistinguishable from $\widetilde{h_i}$; notice that the $\widetilde{h_i}$ are at the lowest depth in the protocol, so we have to use the algorithm **low** on the simulation ciphertexts. A random permutation π is indistinguishable from a random permutation π', and $\{\widehat{k_i}\}_{i\in[m]}$ is a tuple containing encryptions of m random plaintexts $k_i \xleftarrow{\$} \mathbb{M}_F$ with the key pk_L, so, as the LHE scheme is assumed IND-CPA, it is easily simulated with a tuple containing encryptions of m random plaintexts $r'_i \xleftarrow{\$} \mathbb{M}_F$ (using the one-to-one correspondence Ψ). It is clear that $\{\widehat{e_i}\}_{i\in[m]}$ is indistinguishable from $\{\mathbf{L.E}_{pk_L}(\Psi(r_i)) -_L \mathbf{L.E}_{pk_L}(\Psi(r'_i))\}_{i\in[m]}$ as it is constructed the same way with indistinguishable parts. Note that, in the protocol, $|\mathbf{X} \cap \mathbf{Y}|$ of the $\widehat{e_i}$ are encryptions of 0. However, as the LHE scheme is assumed IND-CPA, those encryptions of zeros are indistinguishable from encryptions of anything else. Overall, we have shown that for every subsets $\mathbf{X}, \mathbf{Y} \subset \mathbb{M}_F$:

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

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

$$S_1(\mathbf{X}, (\mathbf{X} \cup \mathbf{Y}, |\mathbf{Y}|)) = \Bigg(\mathbf{X}, \left\{pk_L, sk_L, pk_F, \{r_i\}_{i\in[m]}, \mathbf{F.MEv}(P_{\mathcal{R}}, \{\mathbf{F.E}_{pk_R}(z_i)\}_{i\in[m]}), \{\zeta_{\pi'(i)}\}_{i\in[m]}, \mathcal{D}'\right\},$$
$$\left\{\{\mathbf{F.E}(z_i)\}_{i\in[m]}, \mathbf{F.E}\left(\overleftarrow{P_z}^{-1} \mod Z^{2^{\lceil \log m \rceil}}\right)\right\}, \{\mathcal{T}\}\Bigg).$$

Let $\{z_i\}_{i\in[m]} \subset \mathbb{M}_F$ be a set of $m = |\mathbf{Y}|$ distinct plaintexts, such that exactly $|\mathbf{Y}| - (|\mathbf{X} \cup \mathbf{Y}| - |\mathbf{X}|) = |\mathbf{X} \cap \mathbf{Y}|$ of the z_i (uniformly distributed) are randomly picked in $\mathbf{X}$ and the others are the plaintexts in $\mathbf{X} \cup \mathbf{Y} \setminus \mathbf{X}$.

pk_L, sk_L, pk_F are obtained from the **Setup** algorithm, $P_z := \prod_{i=1}^{|\mathbf{Y}|}(Z - z_i)$, $P_{\mathcal{R}}$ is the plaintext polynomial whose roots are the elements of $\mathbf{X}$, $\pi' \xleftarrow{\$} \mathfrak{S}_m$ is a random permutation of m elements, $\{r_i\}_{i\in[m]}$ is a set of m random plaintexts in $\mathbb{M}_F$, $\zeta_j = \Psi(P_{\mathcal{R}}(z_j) + r_j) - \Psi(r_j)$ for all $j \in [m]$, and $\mathcal{D}'$ and $\mathcal{T}$ depend on the situations presented in Section 4.5 as following. In **Situation 1**,

$$\mathcal{D}' = \{\zeta_{\pi'(i)}, z_{\pi'(i)}\zeta_{\pi'(i)}\}_{i\in[m]}, \tag{109}$$
$$\mathcal{T} = \{\mathbf{L.E}_{pk_L}(\zeta_{\pi'(i)}), \mathbf{L.E}_{pk_L}(z_{\pi'(i)}\zeta_{\pi'(i)})\}_{i\in[m]}. \tag{110}$$

In **Situation 2**,

$$\mathcal{D}' = \{\zeta_{\pi'(i)}, \Psi(z_{\pi'(i)})\zeta_{\pi'(i)}, \Psi(z_{\pi'(i)})(N - \zeta_{\pi'(i)})\}_{i\in[m]}, \tag{111}$$
$$\mathcal{T} = \{\mathbf{L.E}_{pk_L}(\zeta_{\pi'(i)}), \mathbf{L.E}_{pk_L}(\Psi(z_{\pi'(i)})\zeta_{\pi'(i)}), \mathbf{L.E}_{pk_L}(\Psi(z_{\pi'(i)})(N - \zeta_{\pi'(i)}))\}_{i\in[m]}. \tag{112}$$

In **Situation 3**,

$$\mathcal{D}' = \{\mu'_{\pi'(i)}\}_{i\in[m]}, \tag{113}$$
$$\mathcal{T} = \{\mathbf{L.E}_{pk_L}(\zeta_{\pi'(i)})\}_{i\in[m]}, \tag{114}$$

where, for all $i \in [m]$, $\mu'_{\pi'(i)} = \perp$ if $z_i \in \mathbf{X}$, and if not, $\mu'_{\pi'(i)} = z_i$.
With the same arguments than in Appendix A.1, the set $\{z_i\}_{i\in[m]} \subset \mathbb{M}$ is indistinguishable from $\mathbf{Y}$. So by doing everything the same way with the simulated input set $\{z_i\}_{i\in[m]}$, the entire view is indistinguishable. Overall, we obtain for every subsets $\mathbf{X}, \mathbf{Y} \subset \mathbb{M}_F$:

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