

Optimal Communication Unbalanced Private Set Union

Jean-Guillaume Dumas¹, Alexis Galan¹, Bruno Grenet¹, Aude Maignan¹, and Daniel S. Roche²

¹ Univ. Grenoble Alpes, LJK, UMR CNRS 5224, 38000 Grenoble, France
`{firstname.lastname}@univ-grenoble-alpes.fr`

² United States Naval Academy, Annapolis, Maryland, United States
`roche@usna.edu`

Abstract. We present new two-party protocols for the Unbalanced Private Set Union (UPSU) problem. Here, the Sender holds a set of data points, and the Receiver holds another (possibly much larger) set, and they would like for the Receiver to learn the union of the two sets and nothing else. Furthermore, the Sender’s computational cost, along with the communication complexity, should be smaller when the Sender has a smaller set. While the UPSU problem has numerous applications and has seen considerable recent attention in the literature, our protocols are the first where the Sender’s computational cost and communication volume are linear in the size of the Sender’s set only, and do not depend on the size of the Receiver’s set. Our constructions combine linearly homomorphic encryption (LHE) with fully homomorphic encryption (FHE). The first construction uses multi-point polynomial evaluation (MEv) on FHE, and achieves optimal linear cost for the Sender, but has higher quadratic computational cost for the Receiver. In the second construction we explore another trade-off: the Receiver computes fast polynomial Euclidean remainder in FHE while the Sender computes a fast MEv, in LHE only. This reduces the Receiver’s cost to quasi-linear, with a modest increase in the computational cost for the Sender. Preliminary experimental results using HELib indicate that, for example, a Sender holding 1000 elements can complete our first protocol using about 2s of computation time and less than 9MB of communication volume, independently of the Receiver’s set size.

1 Introduction

A private set union (PSU) protocol is a cryptographic protocol involving two parties. The receiver, denoted \mathcal{R} , owns a set \mathbf{X} , and the sender, denoted \mathcal{S} , owns a set \mathbf{Y} . The desired functionality of such a protocol is that the receiver \mathcal{R} receives only the union $\mathbf{X} \cup \mathbf{Y}$, and the sender \mathcal{S} learns nothing. (Note that this is equivalent to the receiver learning the set difference $\mathbf{Y} \setminus \mathbf{X}$.) The protocol is parameterized with (upper bounds on) the set sizes $|\mathbf{X}|$ and $|\mathbf{Y}|$, which are therefore implicitly revealed to both parties as well. However, the sender \mathcal{S} learns nothing about the content of \mathbf{X} and the receiver \mathcal{R} learns nothing about $\mathbf{X} \cap \mathbf{Y}$.

PSU protocols have been widely studied in the case of *balanced* input set size [4, 7, 9, 10, 14–16, 25], motivated by numerous practical applications such as disease data collection from hospitals.

Our interest lies in the *unbalanced* setting, in particular where the sender’s input set is (quite) smaller than the receiver’s. This setting has received less attention, but we note two recent works on UPSU [21, 24] which were developed independently from ours. The motivation here is data *aggregation* from multiple sources which may have different sizes and computational capabilities, and where the set intersections may reveal private relationships, such as IP blacklists [20].

As an illustrative example, consider a whistleblower which has some confidential and compromising data they would like to share with some institution. If the institution obtains some data that was already in its set, it could leak some relationship between different whistleblowers and compromise their anonymity. Similarly, if the whistleblower learns anything, this could tell them about the presence (or not) of other previous whistleblowers. Furthermore, it is expected that one whistleblower’s amount of data as well as their computational resources are smaller than the institution’s, so their computational cost and communication size should be as small as possible; hence UPSU.

If m and n denote the respective sizes of the sets owned by the sender \mathcal{S} and the receiver \mathcal{R} , we thus assume $n \geq m$ and focus on the case where $n \gg m$. The communication volume, that is, the least number of elements exchanged during the protocol, must be $\Omega(m)$ asymptotically, since in the worst case, the whole set of the sender must be sent to the receiver. (This also explains why the other unbalanced case of $m \geq n$ is not promising for improvements.) Also, the optimal arithmetic cost, that is the number of arithmetic operations performed by each party, must be at least linear in the size of their set.

Related Work. In [9] and in [7], the receiver’s set is represented respectively with a polynomial and with a Bloom filter [1] (evaluating to zero exactly in its element’s set). Note that for the Bloom filter case, there is a probability of false-positive, where an element that is not in the receiver’s set evaluates to zero, that has to be controlled with a supplementary statistical security parameter. The representation is sent encrypted under a linearly homomorphic encryption (LHE) [23], and evaluated homomorphically by the sender in all its element’s set. Upon reception and decryption of the evaluations, the receiver obtains either zero and learns nothing, or a non-zero value from which it computes the evaluation point.

In [25], the receiver represents its set \mathbf{X} as a database called an oblivious key-value store (OKVS) [11] in which the elements of \mathbf{X} are viewed as keys, all associated to encryptions of a same secret value. Upon reception, the sender queries the OKVS using its elements as keys. By design, the sender gets encryptions of the secret value for elements in the intersection and random values for elements not in the intersection. Through a sub-protocol, the parties obliviously compare the sender’s values with the receiver’s secret, and the receiver obtains a bit vector where the ones indicate a match. Finally, the parties perform an

oblivious transfer [6] in order for the receiver to learn exactly the elements of \mathbf{Y} that are not in \mathbf{X} .

This protocol has been recently translated into two versions of an UPSU [24]. Both versions use the fact that an OKVS can be structured into a sparse and a dense parts. The sparse part is decodable with low arithmetic cost and communication, and only the small dense part has to be communicated. For efficiency, the first version requires both parties to store their sets into hash tables, with cuckoo hashing for the sender, in order to reduce the union protocol between sets of sizes n and m to about m union protocols between sets of sizes approximately n/m (partitioning). The second version uses a re-randomizable public-key encryption (ReRand-PKE) to skip the sub-protocol step from [25], which compares the secret and the decoded values.

In [21], the global idea is to evaluate a polynomial representing the receiver's set in the sender's elements as in [9], but unlike in the latter, the receiver performs the evaluation. Indeed in an unbalanced context, the large polynomial of the receiver cannot be communicated. To do so, the sender's elements are sent encrypted under FHE, since polynomial evaluation requires both additions and multiplications. Following the ideas from a PSI protocol [5] to reduce the multiplicative depth, the protocol uses batching, windowing, oblivious transfer and partitioning with hash tables. However, as mentioned in [16] and as we show in Section 3, the usage of hash tables to partition the sets, used in [21] and the first protocol of [24], undesirably leaks some information on the intersection set $\mathbf{X} \cap \mathbf{Y}$.

A comparison of the asymptotic complexity bounds of all these protocols and ours are detailed in Table 1. We compare the arithmetic costs for each party and the communication volume, but we also distinguish the protocols through their number of rounds, their compatibility with the security assumptions and, for those that are using fully homomorphic encryption (FHE), as ours, the multiplicative depth of their algorithms, as it has a huge impact on the practical performance. A green value satisfies our goals, while an orange is expected to be improved and a red is not appropriate.

Table 1: Comparison of (U)PSU protocols, where the receiver \mathcal{R} has a set of size n and the sender \mathcal{S} a set of size m , with $n > m$

Protocol	Cost for \mathcal{R}	Cost for \mathcal{S}	Comm. Vol.	# rounds	Depth	Security
[9]	$O(n^{1+\epsilon})$	$O(nm)$	$O(n)$	2		✓
[7]	$O(n)$	$O(m \log n)$	$O(n)$	2		✓
[25]	$O(n)$	$O(m \log n)$	$O(n)$	$\geq 1+OT$		✓
[21]	$O(n)$	$O(m \log n)$	$O(m \log n)$	$\geq 4+OT$	$\leq \log \log(n/m)$	✗
[24, PSU _{op}]	$O(n)$	$O(m \log n)$	$O(m \log n)$	$\geq 6+OT$		✗
[24, PSU _{pk}]	$O(n)$	$O(m \log n)$	$O(m \log n)$	$\geq 2+OT$		✓
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	$2 \log(n/m) + 1$	✓

Our contributions. We present two new UPSU protocols (the second one with a variant), proven secure under the honest-but-curious adversary model.

All our protocols have a communication volume linear in the size of the sender’s set and independent of the size of the receiver’s set.

We also have an optimal number of rounds and a sender’s arithmetic cost which is independent of the receiver’s set size.

Our protocols combine two encryption schemes, namely a linearly homomorphic one and a fully homomorphic one.

We then need to introduce several homomorphic algorithms on polynomials, which can be of independent interest. Our main tool is to use a polynomial representing the receiver’s set (its roots are the receiver’s elements) and evaluate it homomorphically in each of the sender’s elements, but without communicating the whole polynomial.

We optimize here a trade-off between communication volume and FHE multiplicative depth. For security and correctness purposes, we need the FHE scheme to allow exact polynomial evaluation, so we need the plaintext space to be an integral subdomain of the rationals.

The BGV cryptosystem [3] can for instance satisfy this condition as its plaintext space is a finite field.

- Our first protocol is built on fully homomorphic multi-point evaluation, with homomorphic scalar products, low multiplicative depth and large parallelism. We use a few FHE optimizations, namely batching and modulus switching, in order to reduce the time computation and the communication volume. We propose an implementation of this protocol with low communication, using the HELib³ instantiation of BGV.
- Our second protocol relies on efficient fully homomorphic Euclidean remainder and only linearly homomorphic multi-point evaluation. This drastically reduces the arithmetic cost for the receiver, but increases the multiplicative depth and the sender’s cost. Some other trade-offs can be considered and, as a variant, we for instance consider a third protocol, with a slightly worse depth, but a better sender’s cost.

Outline. In Section 2, we introduce the adversary model and the formal security definitions. We then propose in Section 3 a privacy attack on the partitioned constructions with hash tables of [21, 24]. Section 4 contains the linear and fully homomorphic encryption schemes formalization with some practical aspects. From this, in Section 5, we present our optimal communication protocol, using homomorphic batched scalar multi-point evaluation. We also present an implementation with computational timings and communication volume. Finally, in Section 6 we present a protocol (and a variant), using efficient homomorphic algorithms on polynomials, that improve on the asymptotic complexity bounds.

³ <https://github.com/homenc/HELlib>

2 Adversary Model and UPSU Security Definition

Honest-But-Curious Adversaries. Our protocols are secure under the honest-but-curious adversary model, where the participants must follow the protocol but try to learn as much additional information as possible. The security proofs of our protocols, presented in Appendix B, are by simulation, following the framework of [17], where a probabilistic polynomial time (PPT) simulator can generate computationally indistinguishable transcripts [12].

Unbalanced Private Set Union Definition. We propose a formal definition of an UPSU protocol divided into five algorithms, namely **Setup**, **Encode**, **Reduce**, **Map** and **Union**. For a sender \mathcal{S} that owns a set \mathbf{Y} and a receiver \mathcal{R} that owns a set \mathbf{X} :

- $\{keys_{\mathcal{R}}, keys_{\mathcal{S}}\} \leftarrow \mathbf{Setup}(\kappa, \lambda)$: 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 and λ , and outputs $keys_{\mathcal{R}}$ to the receiver and $keys_{\mathcal{S}}$ to the sender;
- $E_{\mathbf{Y}} \leftarrow \mathbf{Encode}(\mathbf{Y}, keys_{\mathcal{S}})$: Given the set \mathbf{Y} and $keys_{\mathcal{S}}$, 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 the set \mathbf{X} , $keys_{\mathcal{R}}$ and $E_{\mathbf{Y}}$. Outputs $R_{\mathbf{X}|E_{\mathbf{Y}}}$ to the sender, an encoding of the set \mathbf{X} , reduced in size depending on $E_{\mathbf{Y}}$;
- $M_{\mathbf{Y}|R_{\mathbf{X}}} \leftarrow \mathbf{Map}(\mathbf{Y}, R_{\mathbf{X}|E_{\mathbf{Y}}}, keys_{\mathcal{S}})$: On input of a set \mathbf{Y} , an encoding $R_{\mathbf{X}|E_{\mathbf{Y}}}$ and $keys_{\mathcal{S}}$, outputs to the receiver an encoded data set $M_{\mathbf{Y}|R_{\mathbf{X}}}$ representing the set $\mathbf{Y} \setminus \mathbf{X}$, depending on \mathbf{Y} and $R_{\mathbf{X}|E_{\mathbf{Y}}}$;
- $\mathbf{Z} \leftarrow \mathbf{Union}(\mathbf{X}, M_{\mathbf{Y}|R_{\mathbf{X}}}, keys_{\mathcal{R}})$: On input of the set \mathbf{X} , an encoded data set $M_{\mathbf{Y}|R_{\mathbf{X}}}$ and $keys_{\mathcal{R}}$, outputs the union set $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$ to the receiver.

Definition 1. (*Setup, Encode, Reduce, Map, Union*) is an unbalanced private set union scheme if it satisfies the following three properties:

1. **Correctness.** For security parameters κ and λ and any sets \mathbf{X}, \mathbf{Y} , for

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

the scheme is correct if

$$\mathbf{Union}(\mathbf{X}, \mathbf{Map}(\mathbf{Y}, R_{\mathbf{X}|E_{\mathbf{Y}}}, keys_{\mathcal{S}}), keys_{\mathcal{R}}) = \mathbf{X} \cup \mathbf{Y}. \quad (1)$$

2. **Privacy.** The scheme is secure under the honest-but-curious adversary model. In particular, the receiver learns $\mathbf{X} \cup \mathbf{Y}$ but nothing about $\mathbf{X} \cap \mathbf{Y}$, and the sender learns nothing.
3. **Unbalanced efficiency.** For input sets \mathbf{X} for the receiver and \mathbf{Y} 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, in particular because usually the receiver sends the first message that has a size proportional to its set size, even if $|\mathbf{Y}| = o(|\mathbf{X}|)$. This justifies the fact that the sender has to send the first message in **Encode**. Then, the receiver uses that message to **Reduce** the representation of its set to a smaller size. A third message is then mandatory for the receiver to get the final information about the union. Up to our knowledge, only [21] and [24] are focused on the unbalanced situation, and their protocols also fit this definition.

3 Polynomial-time Attack on the Partitioning of [21, 24]

We show here that there is a leaky Construction in [21] that is reused in the first protocol of [24] (PSU_{op}). We do not delve into the full constructions, as the leaks actually occur already in the setup phase for both of them. In both protocols indeed, the sender arranges its small set in a cuckoo hash table [8], and the receiver uses the same hash functions to arrange its large set in a simple hash table [19].

A required property of both protocols is that the cuckoo hashing must limit the hash table to have at most one of the sender’s elements per bin. Their protocols, taking as inputs the two sets can now be partitioned into smaller protocols for each bin, taking as inputs the unique sender’s element and the reduced amount of receiver’s elements contained in that bin. This trick is widely used in private set intersection (PSI) protocols for efficiency purposes but, as already mentioned in [16], it cannot be used directly in PSU protocols: there it can leak some clues on the set intersection.

We further show in this section that the leaks increase with the unbalancedness. It is therefore always dangerous to use this trick in UPSU protocols. In the following, the sender \mathcal{S} owns a set \mathbf{Y} of m elements and the receiver \mathcal{R} owns a set \mathbf{X} of n elements.

3.1 Hashing Table Procedure.

For a statistical security parameter λ , \mathcal{S} selects publicly i hash functions $h_j : \{0, 1\}^* \rightarrow [k]$, where $k = k(\lambda) \approx m$ and $i = 3$ for cost efficiency. Those hash functions are chosen such that, with a failure probability below $2^{-\lambda}$, the set \mathbf{Y} can fit in a cuckoo hash table of k bins, without stash. The cuckoo hashing of \mathbf{Y} with h_1, h_2, h_3 then places each element $y \in \mathbf{Y}$ in exactly one bin between $h_1(y)$, $h_2(y)$ or $h_3(y)$, such that at the end of the procedure, each bin contains at most 1 element. On the other side, \mathcal{R} hashes its set \mathbf{X} with a simple hash table, and each element $x \in \mathbf{X}$ is sent in all three bins $h_1(x)$, $h_2(x)$ and $h_3(x)$.

3.2 Leakage on the Intersection.

By definition, a (U)PSU protocol should not leak any clue about the intersection set $\mathbf{X} \cap \mathbf{Y}$ to the receiver. Now, let \mathcal{R} use the three hash functions h_1, h_2, h_3 given by \mathcal{S} , to hash \mathbf{X} in the simple hash table \mathbf{X}_S ; this hash table thus contains $3n$ elements in k bins. If there are 4 distinct elements $x_1, x_2, x_3, x_4 \in \mathbf{X}$ in at most 3 bins $b_1, b_2, b_3 \in [k]$, then the receiver learns that $\{x_1, x_2, x_3, x_4\} \not\subseteq \mathbf{Y}$: indeed, the hash functions are chosen by \mathcal{S} such that each bin in the sender’s cuckoo hash table contains at most 1 element of \mathbf{Y} . Therefore, if $x_1, x_2, x_3, x_4 \in \mathbf{Y}$, then those elements would have been sent in 4 different bins, not 3.

3.3 Modelization.

We consider that the three hash functions are independent and send any element uniformly at random in $[k]$. Let $(b_1, b_2, b_3) \in [k]^3$ be a bin triplet, with distinct b_1, b_2, b_3 . The probability that an element x is sent in bins $\{b_1, b_2, b_3\}$ using one hash function is thus $\frac{3}{k}$. This implies, with independence, that an element x is sent, resp., in bins $\{b_1, b_2, b_3\}$, resp. with h_1, h_2 and h_3 , with probability $p := \left(\frac{3}{k}\right)^3$. If n elements are all sent in bins using h_1, h_2 and h_3 , we want to compute the probability that at least 4 of these elements are sent in $\{b_1, b_2, b_3\}$. We thus model this with a random variable V following a binomial distribution $\mathcal{B}(n, p)$. Now $\mathbb{P}(V \geq 4)$ represents the sought probability with:

$$\mathbb{P}(V \geq 4) = 1 - \sum_{i=0}^3 \binom{n}{i} p^i (1-p)^{n-i}. \tag{2}$$

3.4 Probability of leakage.

In both papers, the statistical security parameter is $\lambda = 40$ and the size of the hash table is $k \approx m + \log m$. In Table 2, we instantiate Eq. (2) for some realistic unbalanced parameters n, m . Table 2 shows that this crude lower bound on the

Table 2: Leaky situation probability lower bounds when $n > m$

n	m	$\mathbb{P}(V \geq 4)$
2^{20}	2^{10}	$\geq 2^{-26}$
2^{10}	10	$\geq 2^{-0,0138} \approx 99.05\%$
2^{20}	10	$\geq 1 - 10^{(-4492)}$

probability of leak is actually way larger than the statistical security parameter, already for a single triplet of bins (while there are in fact $\binom{n}{3}$, not independent, triplets).

3.5 Polynomial-time Attack

To prevent these leaks, one could increase the parameter k , *viz.* $k > 3\sqrt[3]{2^\lambda n}$, but this brings an overhead in communication volume and arithmetic cost for the sender that is no longer sustainable in the unbalanced context. Note also that the leaky situation occurs in fact with probability 1 if $n > 3k^3 + 1$: by the pigeonhole principle there must then be at least one set of 3 bins with 4 distinct elements of \mathbf{X} . This is the root for an always possible polynomial-time attack: in this case, an honest-but-curious attacker with an input set \mathbf{X} , of size n , knowing the three hash functions $h_1, h_2, h_3 : \{0, 1\}^* \rightarrow [k]$, can compute its simple hash table \mathbf{X}_S . The attacker can check if 4 of distinct elements fall in a set of 3 bins. Otherwise, the attacker just adds further distinct elements of its choice in the hash table, until there is a leak (at most $3k^3 + 2 - n$ new known elements have to be added). Thus partitioning is always subject to this attack, requiring only a polynomial number of operations.

In the following, we therefore propose proven secure protocols that do not make use of partitioning.

4 Cryptographic Tools: Homomorphic Schemes

4.1 Linearly Homomorphic Encryption Scheme

Notation: In the following, we denote by \widehat{x} a value encrypted using linearly homomorphic encryption (LHE).

A LHE is a semantically secure public-key encryption scheme such that for $(pk_L, sk_L) \leftarrow \mathbf{L.Setup}(\kappa)$, a key pair for a security parameter κ , together with $\widehat{m}_1 \leftarrow \mathbf{L.E}_{pk_L}(m_1)$ and $\widehat{m}_2 \leftarrow \mathbf{L.E}_{pk_L}(m_2)$, two encryptions, their decryption $\mathbf{L.D}$ must satisfy:

- *homomorphic addition* $+_L$: $\mathbf{L.D}_{sk_L}(\widehat{m}_1 +_L \widehat{m}_2) = m_1 + m_2$.
- *cleartext-ciphertext product* \times_L : $\mathbf{L.D}_{sk_L}(m_1 \times_L \widehat{m}_2) = m_1 m_2$.

For the remainder, we assume that the chosen LHE scheme satisfies IND-CPA security.

Remark 2. We extend naturally the encryption and decryption algorithms for a LHE to allow matrix or polynomials (seen as a vector of its coefficients) as inputs. This allows for instance to extend $+_L$ and \times_L to matrix or polynomial inputs. In particular, we can compute the homomorphic polynomial product between a cleartext polynomial A and a ciphertext polynomial \widehat{C} .

4.2 Fully Homomorphic Encryption Scheme

Notation: In the following, we denote by \widetilde{x} a value encrypted using fully homomorphic encryption (FHE).

A FHE is a semantically secure public-key encryption scheme such that for $(pk_F, sk_F) \leftarrow \mathbf{F.Setup}(\kappa)$, a key pair for a security parameter κ , together with $\widetilde{m}_1 \leftarrow \mathbf{F.E}_{pk_F}(m_1)$ and $\widetilde{m}_2 \leftarrow \mathbf{F.E}_{pk_F}(m_2)$, two encryptions, their decryption $\mathbf{F.D}$ must satisfy:

- *homomorphic addition* $+_F$: $\mathbf{F.D}_{sk_F}(\widetilde{m}_1 +_F \widetilde{m}_2) = m_1 + m_2$.
- *cleartext-ciphertext product* \times_F : $\mathbf{L.D}_{sk_F}(m_1 \times_F \widetilde{m}_2) = m_1 m_2$.
- *homomorphic product* \times_F : $\mathbf{F.D}_{sk_F}(\widetilde{m}_1 \times_F \widetilde{m}_2) = m_1 m_2$.

Remark 3. Similarly to the LHE, we extend naturally the scheme for matrix and polynomial inputs, and we assume for the purpose of our proofs that the FHE is IND-CPA secure.

4.3 Practical Tools in FHE.

In practice, actual FHE schemes, like the BGV scheme [3], are constructed on top of a leveled fully homomorphic encryption scheme, allowing a bounded multiplicative depth algorithms. The latter is extended to arbitrary depth using a *bootstrapping* procedure. A message is encrypted with a random noise that grows at each homomorphic operation. If the noise becomes too large, the ciphertext is no longer decipherable. Before that, a *modulus switching* procedure can be performed (in particular, after each homomorphic product), reducing the noise (and the size of the ciphertext). When the ciphertext size is even too small for another modulus switching, then only a bootstrapping is required (increasing back the size of the ciphertext, while regaining a small noise). Bootstrapping being usually quite expensive, it means that the multiplicative depth of the protocols has to be controlled. We further introduce two tools that we use in practice in our protocols. The first one is a *noise flooding* which, together with a *shortening*, allows to ensure *circuit privacy*, while incidentally reducing the communication volume. The second tool is *batching*, that allows single-instruction multiple-data (SIMD) homomorphic operations.

Noise Flooding and Shortening In some implementations of FHE, ciphertexts size and noise could contain some information about the homomorphic operation performed. To prevent this, and thus ensure circuit privacy, some schemes often need another algorithm, called **flood**, that, given an FHE ciphertext \widetilde{y} , performs a noise flooding⁴ together with as many modulus switching as possible (to preserve the decryption's correctness). The resulting ciphertext encrypts the same cleartext, but can no more be multiplied homomorphically without a bootstrap. Now, for f, g two arithmetic circuits, let a, b, y be such that $f(a) = g(b) = y$ and let $\widetilde{a}, \widetilde{b}$ be their respective FHE encryptions, the decryption key owner should then not be able to distinguish $\mathbf{flood}(f(\widetilde{a}))$ from $\mathbf{flood}(g(\widetilde{b}))$.

Batching From the construction of cleartext and ciphertext spaces in FHE schemes, it is often possible to batch together several cleartexts so that encrypting them will result in a single ciphertext. We will denote by $\langle y_i \rangle_{i \in [m]}$ a batch of m cleartexts. For $\widetilde{\mathbf{y}} \leftarrow \mathbf{F.E}_{pk_F}(\langle y_i \rangle_{i \in [m]})$ an encryption of batched cleartexts and f a circuit, the batching correctness implies that $f(\widetilde{\mathbf{y}})$ is an encryption of $\langle f(y_i) \rangle_{i \in [m]}$.

⁴ For instance, by adding a random noise.

5 Optimal Communication Volume, Low Depth, Batchable and Parallelizable UPSU Protocol

In our protocol, we represent the set \mathbf{X} owned by the receiver as the polynomial $P_{\mathcal{R}}(Z) = \prod_{x \in \mathbf{X}} (Z - x)$ and we evaluate $P_{\mathcal{R}}$ on each element owned by the sender. The receiver must learn only the elements of \mathbf{Y} that are not a root of $P_{\mathcal{R}}$, and the evaluation must be performed homomorphically. The idea to keep the communication volume proportional to the size $|\mathbf{Y}|$ of the sender's set, is that the sender first sends its elements, encrypted. The receiver performs the evaluation homomorphically on its polynomial in clear. This requires both additions and multiplications, whence the need for a FHE scheme. For efficiency purposes, we introduce a homomorphic polynomial evaluation algorithm that is highly parallelizable, uses properly the batch and has a low multiplicative depth. Finally, to minimize the number of rounds and avoid the expensive usage of FHE when it is not required, we make a transition from the FHE to the LHE scheme and conclude our protocol similarly to [9].

5.1 Fully Homomorphic Batched Scalar Multi-point Evaluation

For a pair of FHE keys (pk_F, sk_F) , a homomorphic batched scalar multi-point evaluation algorithm, denoted $\mathbf{F.BSMEv}$, is an algorithm that, given a batched ciphertext $\tilde{\mathbf{y}} \leftarrow \mathbf{F.E}_{pk_F}(\langle y_i \rangle_{i \in [m]})$ and a polynomial P in clear, satisfies

$$\mathbf{F.D}_{sk_F}(\mathbf{F.BSMEv}(P, \tilde{\mathbf{y}})) = \langle P(y_i) \rangle_{i \in [m]}. \quad (3)$$

This algorithm evaluates a clear polynomial in many encrypted evaluation points. Its main goal is to be efficient in practice. This requires to use properly the batching, to have a low multiplicative depth, to lessen the number of homomorphic products and to be as parallelizable as possible.

Proposition 1. *Let P be a clear polynomial of degree n given as a product of \sqrt{n} polynomials of degrees \sqrt{n} , and let $\tilde{\mathbf{y}} \leftarrow \mathbf{F.E}_{pk_F}(\langle y_i \rangle_{i \in [m]})$ be a batched ciphertext corresponding to m plaintexts. Then $\mathbf{F.BSMEv}(P, \tilde{\mathbf{y}})$ can be computed in $O(mn)$ arithmetic operations and a depth $\lceil \log n \rceil + 1$.*

Proof. The algorithm is built in three steps. First, compute homomorphically the encryption of y_i^j , for $i \in [m]$ and $j \in [\sqrt{n}]$. Using batching, this costs $m\sqrt{n}$ homomorphic products \times_F , in depth $\log \sqrt{n}$. The result is an encrypted vector $\vec{\mathbf{v}}$. Let $P = P_1 \times \dots \times P_{\sqrt{n}}$, with $\deg P_j = \sqrt{n}$. Then, write each P_j as a vector of coefficients and compute the encryptions $\widetilde{P_j(\mathbf{y})}$ of $\langle P_j(y_i) \rangle_{i \in [m]}$, for $j \in [\sqrt{n}]$, using homomorphic inner products between the vectors of coefficients and $\vec{\mathbf{v}}$. This requires nm cleartext-ciphertext products \times_F , in multiplicative depth 1. Finally, reconstruct homomorphically the encryption of $\langle P(y_i) \rangle_{i \in [m]}$ from $\{\widetilde{P_j(\mathbf{y})}\}_{j \in [\sqrt{n}]}$ with a (homomorphic) binary multiplicative tree. This requires $m\sqrt{n}$ homomorphic products \times_F in depth $\log \sqrt{n}$.

5.2 Formalization of the protocol

Let $\{x_i\}_{i \in [n]}$, $\{y_i\}_{i \in [m]} \subset \mathbb{M}$, $n \geq m$, be the sets of the receiver \mathcal{R} and of the sender \mathcal{S} , respectively. For the sake of simplicity, we assume that \mathbb{M} is a finite field, and that it is the (common) plaintext space of an FHE and an LHE. This assumption is met in our implementations, and we propose in Appendix A some slight changes to keep the correctness of our protocol under other assumptions. The respective ciphertext spaces for LHE and FHE are denoted \mathbb{E}_L and \mathbb{E}_F . 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.

Theorem 1. *The protocol built with the algorithms **Setup**, **Encode**, **Reduce**, **Map** and **Union** (Algs. 1 to 5) is a secure unbalanced private set union scheme under the honest-but-curious adversary model and 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 correctness of the scheme relies on the fact that an element y owned by the sender has to be added to the receiver’s set \mathbf{X} if, and only if, $P_{\mathcal{R}}(y) \neq 0$. The correctness of the evaluations is implied by the correctness of the FHE and LHE encryption schemes, as well as the fact that both scheme share the same field as plaintext space. The security proof, using simulation, is presented in Appendix B.

5.3 Timings in HELib

To instantiate Protocol 1, we used the C++ open source library HELib which implements the BGV cryptosystem [3]; the source code is available at <https://github.com/GalanAI/CoUPSU>. We could have done our implementations on any other library implementing exact FHE schemes (BGV, BFV, ...), as SEAL⁵ and OpenFHE⁶, but we found more freedom in the choice of parameters with

⁵ <https://github.com/microsoft/SEAL>

⁶ <https://github.com/openfheorg/openfhe-development>

Algorithm 1 Setup(κ)

Input: A security parameter κ .**Outputs:** Pairs of LHE keys (pk_L, sk_L) and FHE keys (pk_F, sk_F) , both with κ bits of security. The secret key sk_L is owned by the receiver, and sk_F is owned by the sender.

- 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 plaintexts $\mathbf{Y} = \{y_i\}_{i \in [m]} \subset \mathbb{M}$ and $keys_{\mathcal{S}} = \{(pk_F, sk_F), pk_L\}$.**Output:** A FHE ciphertext $\tilde{\mathbf{y}} \in \mathbb{E}_F$, such that $\mathbf{F.D}_{sk_F}(\tilde{\mathbf{y}}) = \langle y_i \rangle_{i \in [m]}$.

- 1: \mathcal{S} : compute $\tilde{\mathbf{y}} \leftarrow \mathbf{F.E}_{pk_F}(\langle y_i \rangle_{i \in [m]})$ and send $\tilde{\mathbf{y}}$ to \mathcal{R} ; ▷ Batch and encrypt
 - 2: \mathcal{R} : **return** $\tilde{\mathbf{y}}$;
-

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

Input: A set of plaintexts $\mathbf{X} = \{x_j\}_{j \in [n]} \subset \mathbb{M}$, a FHE ciphertext $\tilde{\mathbf{y}} \in \mathbb{E}_F$ and $keys_{\mathcal{R}} = \{(pk_L, sk_L), pk_F\}$.**Output:** A reduced FHE ciphertext $\tilde{\mathbf{h}} \in \mathbb{E}_F$, and a set of LHE ciphertexts $\{\hat{k}_i\}_{i \in [m]} \subset \mathbb{E}_L$, such that, for $\mathbf{F.D}_{sk_F}(\tilde{\mathbf{y}}) = \langle y_i \rangle_{i \in [m]}$,

$$\mathbf{F.D}_{sk_F}(\tilde{\mathbf{h}}) = \langle \mathbf{L.D}_{sk_L}(\hat{k}_i) + \prod_{j \in [n]} (y_i - x_j) \rangle_{i \in [m]}.$$

- 1: \mathcal{R} : compute $P_{\mathcal{R}} \leftarrow \prod_{j \in [n]} (Z - x_j)$;
 - 2: **for all** $i \in [m]$ **do**
 - 3: \mathcal{R} : sample $k_i \xleftarrow{\$} \mathbb{M}$ uniformly;
 - 4: \mathcal{R} : compute $\hat{k}_i \leftarrow \mathbf{L.E}_{pk_L}(k_i)$; ▷ Encrypt the masks under LHE
 - 5: **end for**
 - 6: \mathcal{R} : compute $\tilde{\mathbf{k}} \leftarrow \mathbf{F.E}_{pk_F}(\langle k_i \rangle_{i \in [m]})$; ▷ Encrypt the batched masks under FHE
 - 7: \mathcal{R} : compute $\tilde{\mathbf{e}} \leftarrow \mathbf{F.BSMEv}(P_{\mathcal{R}}, \tilde{\mathbf{y}})$; ▷ Homom. multi-point evaluation
 - 8: \mathcal{R} : compute $\tilde{\mathbf{h}} \leftarrow \mathbf{flood}(\tilde{\mathbf{k}} +_F \tilde{\mathbf{e}})$; ▷ Mask, reduce and flood the evaluations
 - 9: \mathcal{R} : send $\tilde{\mathbf{h}}$ and $\{\hat{k}_i\}_{i \in [m]}$ to \mathcal{S} ;
 - 10: \mathcal{S} : **return** $\{\tilde{\mathbf{h}}, \{\hat{k}_i\}_{i \in [m]}\}$;
-

Algorithm 4 $\text{Map}(\mathbf{Y}, \{\widehat{\mathbf{h}}, \{\widehat{k}_i\}_{i \in [m]}\}, \text{keys}_S)$

Input: A set of m plaintexts $\mathbf{Y} = \{y_i\}_{i \in [m]} \subset \mathbb{M}$, a FHE ciphertext $\widetilde{\mathbf{h}} \in \mathbb{E}_F$, a LHE ciphertext set $\{\widehat{k}_i\}_{i \in [m]} \subset \mathbb{E}_L$, and $\text{keys}_S = \{(pk_F, sk_F), pk_L\}$.

Output: A set of LHE ciphertext pairs $\{(\widehat{e}_j, \widehat{\eta}_j)\}_{j \in [m]} \subset \mathbb{E}_L^2$ such that, for

$\mathbf{F.D}_{sk_F}(\widetilde{\mathbf{h}}) = \langle h_i \rangle_{i \in [m]}$, $\mathbf{L.D}_{sk_L}(\widehat{e}_{\pi(i)}) = h_i - \mathbf{L.D}_{sk_L}(\widehat{k}_i)$ and

$\mathbf{L.D}_{sk_L}(\widehat{\eta}_{\pi(i)}) = y_i \mathbf{L.D}_{sk_L}(\widehat{e}_{\pi(i)})$ for $i \in [m]$.

- 1: \mathcal{S} : compute $\langle h_i \rangle_{i \in [m]} \leftarrow \mathbf{F.D}_{sk_F}(\widetilde{\mathbf{h}})$; ▷ Decrypt the masked evaluations
 - 2: \mathcal{S} : sample a uniform permutation $\pi \xleftarrow{\$} \mathfrak{S}_m$;
 - 3: **for all** $i \in [m]$ **do**
 - 4: \mathcal{S} : compute $\widehat{e}_{\pi(i)} \leftarrow \mathbf{L.E}_{pk_L}(h_i) -_L \widehat{k}_i$; ▷ Remove the masks under LHE
 - 5: \mathcal{S} : compute $\widehat{\eta}_{\pi(i)} \leftarrow y_i \times_L \widehat{e}_{\pi(i)}$; ▷ Multiply by the evaluation point
 - 6: **end for**
 - 7: \mathcal{S} : send $\{(\widehat{e}_{\pi(i)}, \widehat{\eta}_{\pi(i)})\}_{i \in [m]}$ to \mathcal{R} ;
 - 8: \mathcal{R} : **return** $\{(\widehat{e}_j, \widehat{\eta}_j)\}_{j \in [m]}$;
-

Algorithm 5 $\text{Union}(\mathbf{X}, \{(\widehat{e}_j, \widehat{\eta}_j)\}_{j \in [m]}, \text{keys}_{\mathcal{R}})$

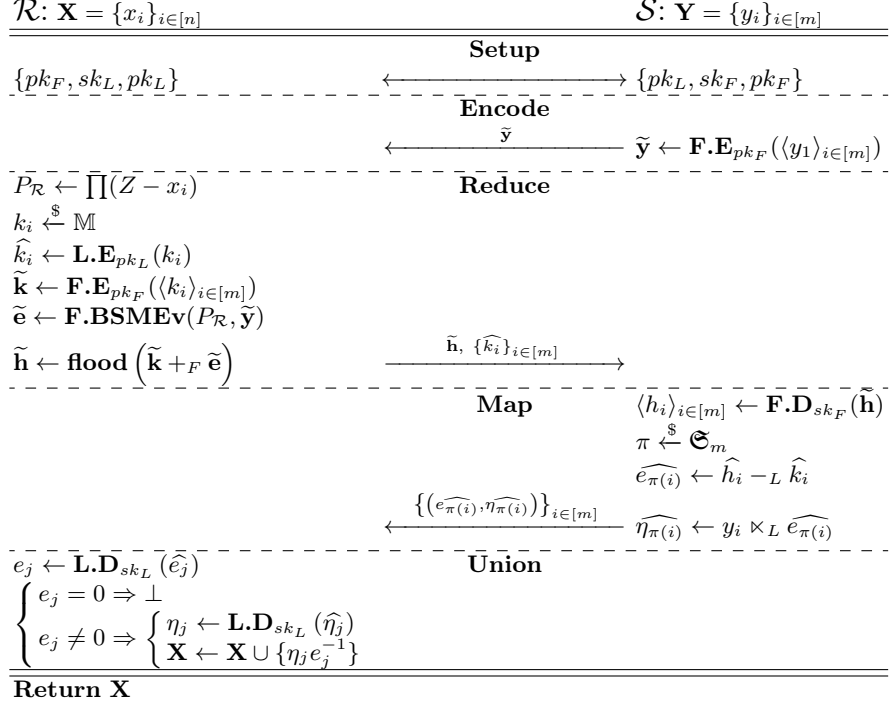
Input: A set of plaintexts $\mathbf{X} = \{x_j\}_{j \in [m]} \subset \mathbb{M}$, a set of LHE ciphertext pairs $\{(\widehat{e}_j, \widehat{\eta}_j)\}_{j \in [m]} \subset \mathbb{E}_L^2$.

Output: A set of plaintexts $\mathbf{Z} \subset \mathbb{M}$, such that

$\mathbf{Z} = \mathbf{X} \cup \{\mathbf{L.D}_{sk_L}(\widehat{\eta}_j) \mathbf{L.D}_{sk_L}(\widehat{e}_j)^{-1} : \mathbf{L.D}_{sk_L} \neq 0\}$.

- 1: \mathcal{R} : compute $\mathbf{Z} \leftarrow \mathbf{X}$;
 - 2: **for all** $j \in [m]$ **do**
 - 3: \mathcal{R} : compute $e_j \leftarrow \mathbf{L.D}_{sk_L}(\widehat{e}_j)$;
 - 4: **if** $e_j \neq 0$ **then**
 - 5: \mathcal{R} : compute $\mathbf{Z} \leftarrow \mathbf{Z} + \{\mathbf{L.D}_{sk_L}(\widehat{\eta}_j) e_j^{-1}\}$; ▷ Add the non-roots to the set
 - 6: **end if**
 - 7: **end for**
 - 8: \mathcal{R} : **return** \mathbf{Z} ;
-

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



HElib. We used a Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz with 56 threads for our experiments. The plaintext space is the field \mathbb{F}_{614332} , that contains 384 bit-length words. The ciphertext space allows to batch up to 1024 plaintexts 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. We choose the other parameters of the context in order to keep the correctness of the decryption after an homomorphic circuit of multiplicative depth 21. Through the security estimation given by HElib, we have $\kappa = 115$.

In our implementation, we use BGV restricted to linear operations as a LHE. Since the receiver is assumed to have significant computing power, we use parallelization on all the available threads, mainly for $\mathbf{F} \cdot \mathbf{BSMEv}$ in **Reduce**. We ignore the communication volume and the runtime implied by **Setup** as it can be done offline, and consider that both parties own a secret key and the two public keys. In the unbalanced situation, it is interesting to distinguish the runtime of both parties, presented in Fig. 1a and Fig. 1b respectively.

The experimental results confirm the expected asymptotic presented in Table 3. In particular, the arithmetic cost for the sender is independent of the size

of the receiver’s set, and is low. On the receiver’s side, the runtime values confirm the linear complexity in the size of its set. The runtime for the receiver is however quite large, around 5800 seconds for a set of size 2^{22} , but the implementation could be optimized to reduce it. Also, the algorithm is highly parallelizable and the receiver is assumed to have a significant computing power, so the runtime may also be decreased by using more threads. Fig. 1c confirms that the communication volume is independent of the receiver’s set size and we compare it to the values given in [21, 24, 25].⁷ We observe that due to some optimizations, for example the leaky partitioning of [21], the communication volume of previous protocols are smaller than ours when the receiver’s set is small. Yet, for larger sets we obtain better results thanks to the optimal asymptotic volume of our protocol.

6 Asymptotic Improvement for the Receiver

In this section, the idea is to use efficient computer algebra algorithms adapted to the homomorphic encryption constraints. This allows us to improve the arithmetic cost for the receiver, making it quasi-linear in its – large – set size n . We are able to do this while keeping an optimal communication volume, $O(m)$, as well as 3 rounds and a sender’s arithmetic cost independent of the receiver’s set size (quasi-linear in its – small – set size). The downside is that batching is now more complex to set-up and that the multiplicative depth is doubled, making their implementations currently less efficient on the examples of Section 5

More precisely, we reduce the arithmetic costs of our algorithms to fast polynomial multiplication, between two ciphertext polynomials, one cleartext and one ciphertext, or two cleartext polynomials. Even if fast polynomial multiplication is quasi-linear in the degree, the context may vastly modify the actual constant in the complexity bound. We thus detail in the following the number of polynomial products in each context, for A, B polynomials of degrees at most d :

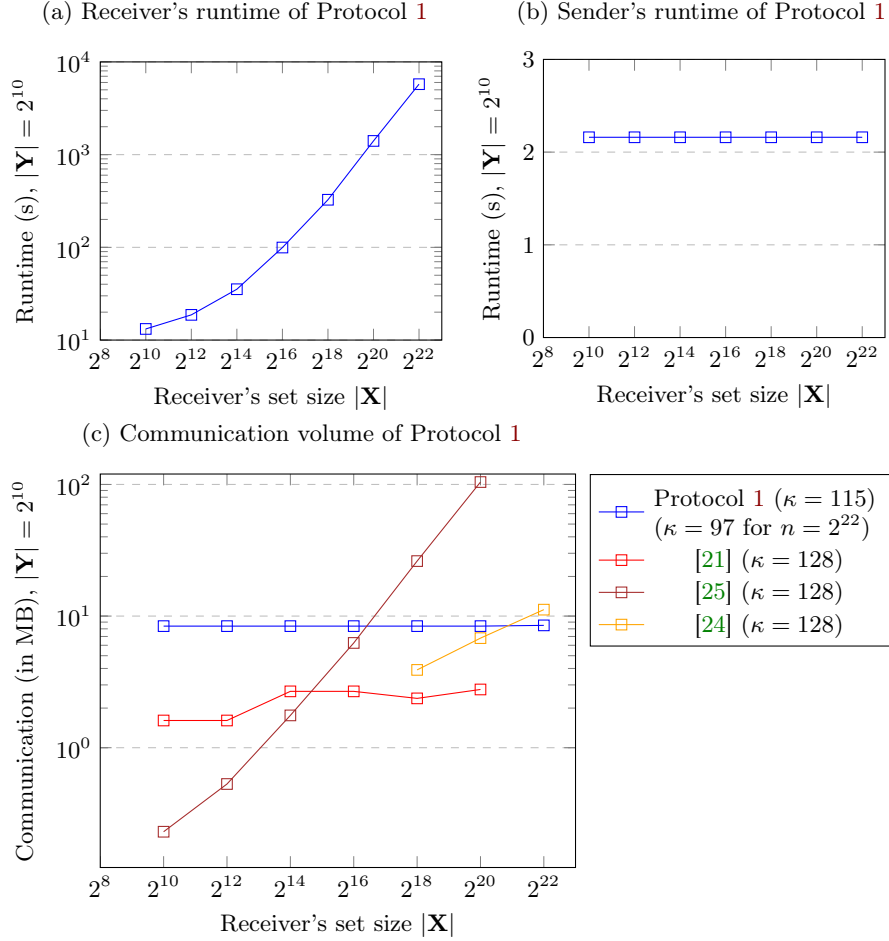
- $\mathcal{M}(d)$ is a bound on the arithmetic cost of the map $A, B \mapsto AB$;
- $\mathcal{M}_F(d)$ is a bound on the arithmetic cost of the map $\tilde{A}, \tilde{B} \mapsto \tilde{A} \times_F \tilde{B}$ (note that a homomorphic polynomial product is considered to have a multiplicative depth 1 in that paper);
- $\mathcal{M}_L(d)$ is a bound on the arithmetic cost of the map $A, \hat{B} \mapsto A \times_L \hat{B}$.

6.1 Efficient LHE and FHE Fast Multi-point Evaluation

We consider $P_{\mathcal{R}}$ and $P_{\mathcal{S}}$, the two polynomials whose roots are respectively the elements of the sets \mathbf{X} and \mathbf{Y} . To avoid the set intersection, we want to discard the points of \mathbf{Y} that evaluate $P_{\mathcal{R}}$ to zero. Our first remark is that it is sufficient to use the remainder $R = P_{\mathcal{R}} \bmod P_{\mathcal{S}}$ for the evaluations. But now this polynomial has a degree at most $m - 1$ for $m = |\mathbf{Y}|$. Our idea, in Protocol 2, is then to

⁷ In the other papers, the element bit-length is 128, but it is 384 in our implementations.

Fig. 1: Protocol 1 experimental results



homomorphically compute and evaluate this remainder instead. We show how to preserve a fast arithmetic complexity bound, even under LHE or FHE.

Efficient Fully Homomorphic Euclidean Remainder. In the context of homomorphic encryption, the impossibility of branching, for instance, makes it more complex to design a *fast* homomorphic remainder. The classical fast division algorithm computes the quotient first by a fast modular inverse of the divisor and then updates the remainder. The fast modular inverse is usually obtained by a Newton-like iteration. Here we need the division of a clear polynomial by a ciphered one. Even then, this computation cannot be directly 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 also be an issue. We thus focus here on the case where the divisor is monic. We denote this algorithm with the operator \mathbf{mod}_F , that, for $(pk_F, sk_F) \leftarrow \mathbf{F.Setup}(\kappa)$, a FHE key pair of security κ and for two cleartext polynomials A, B with B monic, do satisfy that:

$$\mathbf{F.D}_{sk_F}(A \mathbf{mod}_F(\mathbf{F.E}_{pk_F}(B))) = A \pmod{B}. \quad (4)$$

Proposition 2. *Let A be a cleartext polynomial of degree n and let \tilde{B} be a FHE ciphertext polynomial which is an encryption of a monic polynomial of degree $m < n$. A $\mathbf{mod}_F(\tilde{B})$ can be computed in less than $4\mathcal{M}_F(n - m) + O(n)$ arithmetic operations with a depth $2\lceil \log(n - m + 1) \rceil + 1$. If an encryption of $\overleftarrow{\tilde{B}}^{-1} \pmod{Z^{2^{\lceil \log m \rceil}}}$ is given, the depth is reduced to $2\lceil \log(n - m + 1) - \log m \rceil + 1$.*

Proof. The algorithm presented in Section C.1 is an adaptation of the fast Euclidean division algorithm, based on Newton iteration [22], to the FHE settings. We remark that if an encryption of $\overleftarrow{\tilde{B}}^{-1} \pmod{Z^{2^{\lceil \log m \rceil}}}$ is given, where $\overleftarrow{\tilde{B}}$ is the reverse polynomial, the first $\lceil \log m \rceil$ steps of the Newton iterations can be skipped, reducing the multiplicative depth by $2\lceil \log m \rceil$.

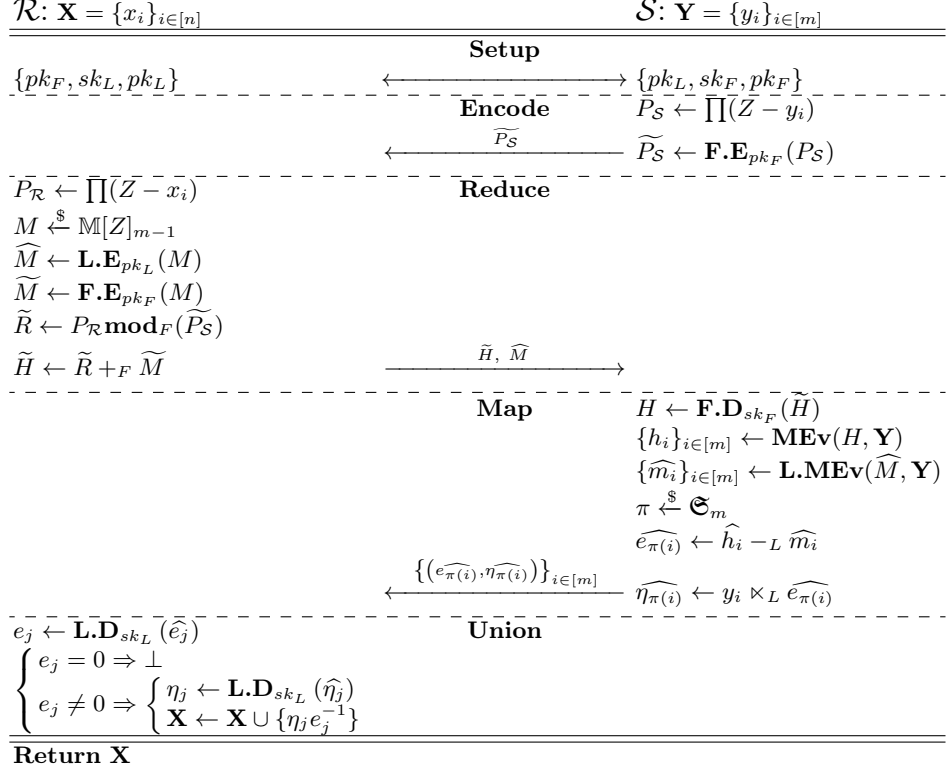
Fast Linearly Homomorphic Multi-point Evaluation. We define a linearly homomorphic multi-point polynomial evaluation algorithm, denoted by $\mathbf{L.MEv}$, that homomorphically evaluates a ciphertext polynomial of degree $m - 1$ in m cleartext evaluation points. For $(pk_L, sk_L) \leftarrow \mathbf{L.Setup}(\kappa)$, a LHE key pair of security κ , for H a cleartext polynomial of degree $m - 1$ and for m cleartext evaluation points $\{y_i\}_{i \in [m]}$, if $\{\hat{e}_i\}_{i \in [m]} \leftarrow \mathbf{L.MEv}(\mathbf{L.E}_{pk_L}(H), \{y_i\}_{i \in [m]})$ then, this algorithm has to satisfy, for all $i \in [m]$, that: $\mathbf{L.D}_{sk}(\hat{e}_i) = H(y_i)$. Naively, the algorithm $\mathbf{L.MEv}$ can be implemented in $O(m^2)$ operations, evaluating homomorphically the polynomial on each point with an adaptation of the Horner scheme. In Section C.2 we instead derive from [2] an asymptotically fast multi-point evaluation algorithm that satisfies:

Proposition 3. *Let \hat{H} be an LHE ciphertext polynomial of degree $m - 1$ and $\{y_i\}_{i \in [m]}$ be m cleartext evaluation points. With $\frac{1}{2}\mathcal{M}(m) \log m + O(m)$ operations of precomputation on the y_i , $\mathbf{L.MEv}(\hat{H}, \{y_i\}_{i \in [m]})$ can be computed in at most $\mathcal{M}_L(m) \log m + \tilde{O}(m)$ operations.*

Protocol with \mathbf{mod}_F and $\mathbf{L.MEv}$. Our second protocol makes use of two cryptosystems, a LHE and a FHE. To ensure correctness, the schemes have to share the same cleartext space. It is presented in Protocol 2 and its correctness is implied by the correctness of the encryption schemes and the fact that the remainder $P_{\mathcal{R}} \pmod{P_{\mathcal{S}}}$ vanishes only on the intersection set.

The full simulation proof of the security of Protocol 2 is given in Appendix B. The asymptotic presented in Table 4 are implied by Prop. 3 and Prop. 2. In particular, the sender sends $\overleftarrow{P_{\mathcal{S}}}$ together with an encryption of $\overleftarrow{P_{\mathcal{S}}}^{-1} \pmod{Z^{2^{\lceil \log m \rceil}}}$

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



in its first message, to reduce the FHE multiplicative depth of $P_{\mathcal{R}} \mathbf{mod}_F(\widetilde{P}_S)$, without increasing the dominant term of the communication volume. We observe that each party has a quasi-linear arithmetic cost in its set size, while conserving an optimal communication volume. Also, we can remark that this protocol has a lower multiplicative depth than Protocol 1 when $\sqrt{n} < m < n$, even if it is not our target usage case.

Theorem 2. *Protocol 2 is a secure UPSU under the honest-but-curious adversary model. For the receiver and the server respectively owning sets \mathbf{X} and \mathbf{Y} of n and m cleartexts, with the assumption that $n > m$, it computes the set union with the asymptotic complexity bounds presented in Table 4.*

6.2 Fast FHE Multi-point Evaluation with Compatibility LHE-FHE

In fact, as long as the divisor is monic, we show in the following that it is actually possible to perform a fast multi-point evaluation, based on successive remainders, from a FHE Euclidean remainder. This is the idea of the variant

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	$O(1)$	$\tilde{O}(m)$	$\tilde{O}(m)$	
Reduce	$4\mathcal{M}_F(n) + \tilde{O}(n)$	$O(1)$	$O(m)$	$2(\lceil \log(n - m + 1) \rceil - \lfloor \log m \rfloor) + 1$
Map	$O(1)$	$\mathcal{M}_L(m) \log m + \tilde{O}(m)$	$O(m)$	
Union	$O(m)$	$\tilde{O}(1)$	$O(1)$	
Total	$4\mathcal{M}_F(n) + \tilde{O}(n)$	$\mathcal{M}_L(m) \log m + \tilde{O}(m)$	$O(m)$	$2(\lceil \log(n - m + 1) \rceil - \lfloor \log m \rfloor) + 1$

version presented in Protocol 3. This protocol allows a quasi-linear arithmetic cost for both parties, in their respective set size and an optimal communication volume. It is clearer to present it when the LHE and the FHE have the same cleartext space, but we show in Appendix A that this condition is not mandatory. The algorithm can be parallelized but batching is more difficult, and its depth is twice that of Protocol 1. In comparison to Protocol 2, the cost for \mathcal{S} is reduced, as it does not need to perform the homomorphic multi-point evaluation, but the multiplicative depth is increased by $2 \log m$.

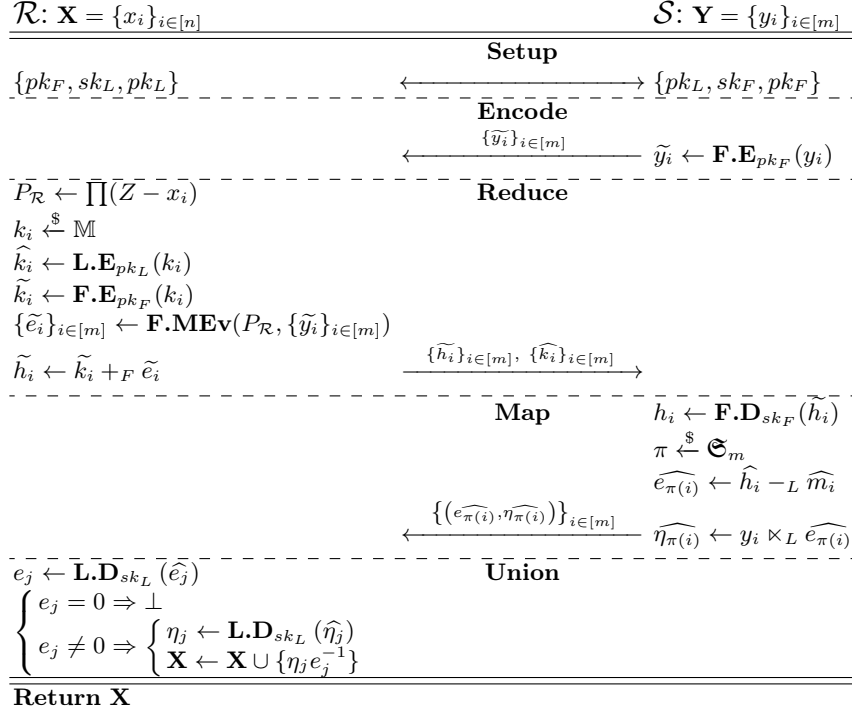
Fast Fully Homomorphic Multi-point Evaluation. We thus now denote by **F.MEv**, a fast FHE multi-point evaluation algorithm, that homomorphically evaluates a cleartext polynomial in encrypted evaluation points. For $(pk_F, sk_F) \leftarrow \mathbf{F.Setup}(\kappa)$, a FHE key pair of security κ , for a cleartext polynomial A and for m cleartext evaluation points $\{y_i\}_{i \in [m]}$, if $\{\tilde{e}_i\}_{i \in [m]} \leftarrow \mathbf{F.MEv}(A, \{\mathbf{F.E}_{pk_F}(y_i)\}_{i \in [m]})$ then, this algorithm has to satisfy, for all $i \in [m]$, that: $\mathbf{F.D}_{sk_F}(\tilde{e}_i) = A(y_i)$. We show in Section C.3 that it is possible to adapt a Newton iterations and a fast multi-point algorithm to the fully homomorphic context. This allows us to get the following proposition:

Proposition 4. *Let A be of degree n and $\{\tilde{y}_i\}_{i \in [m]}$ be $m < n$ ciphertexts. $\mathbf{F.MEv}(A, \{\tilde{y}_i\}_{i \in [m]})$ can be computed in less than $4\mathcal{M}_F(n - m) + O(n)$ arithmetic operations with a depth $2(L + l)$, for $L = \lceil \log(n - m + 1) \rceil$ and $l = \lceil \log m \rceil$. If an encryption of $\prod_{i \in [m]} (Z - y_i)^{-1} \bmod Z^{2^l}$ is given, the depth is reduced to $2L$.*

Proof. The algorithm presented in Section C.3 is a modification of the Newton iterations and a multi-point algorithm based on successive Euclidean remainders adapted in the FHE context. Similarly to Protocol 2, having an encryption of $\prod_{i \in [m]} (Z - y_i)^{-1} \bmod Z^{2^l}$ allows to skip l steps of the Newton iterations, and reduce the multiplicative depth by $2l$.

Protocol with F.MEv. This allows us to present Protocol 3 whose characteristics are given in Theorem 3.

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



The correctness proof of Protocol 3 follows the line of that of Protocol 1 and the simulator is given in Appendix B. Its asymptotic cost is presented in Table 5 and is implied by Prop. 4. In particular, the sender sends $\widehat{P}_{\mathcal{S}}$ together with an encryption of $\prod_{i \in [m]} (Z - y_i)^{-1} \pmod{Z^{2^{\lceil \log m \rceil}}}$ in its first message. This reduces the FHE multiplicative depth of $\mathbf{F.MEv}(P_{\mathcal{R}}, \{\tilde{y}_i\}_{i \in [m]})$, without increasing the dominant term of the communication volume. We have proven:

Theorem 3. *Protocol 3 is a secure UPSU under the honest-but-curious adversary model. For the receiver and the server respectively owning sets \mathbf{X} and \mathbf{Y} of n and m cleartexts, with the assumption that $n > m$, it computes the set union with the asymptotic complexity bounds presented in Table 5.*

Remark 4. According to Prop. 4, the sender's arithmetic cost can be reduced to be linear in its set size, but this would lead to an increase of the multiplicative depth by $2 \log m$.

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	$O(1)$	$\tilde{O}(m)$	$\tilde{O}(m)$	
Reduce	$4\mathcal{M}_F(n) + \tilde{O}(n)$	$\tilde{O}(1)$	$\tilde{O}(m)$	$2 \log(n - m + 1)$
Map	$\tilde{O}(1)$	$\tilde{O}(m)$	$\tilde{O}(m)$	
Union	$\tilde{O}(m)$	$\tilde{O}(1)$	$\tilde{O}(1)$	
Total	$4\mathcal{M}_F(n) + O(n)$	$\tilde{O}(m)$	$O(m)$	$2 \log(n - m + 1)$

7 Conclusion

We have presented two new protocols for the UPSU problem, which are the first whose computational and communication costs for the sender depend only on the size of the sender’s set. Asymptotically, we have achieved the optimal or nearly-optimal computational and communication costs.

However, our protocols are still largely of theoretic interest and further work remains to make them viable in practice. Our preliminary experiments for the first protocol validate our asymptotics and show that the sender computation time and communication volume are competitive with other recent UPSU work. However, we would need to scale this to larger sizes (particularly larger *receiver* set sizes) to demonstrate the practical advantage over alternative approaches.

This scaling is hindered by the higher receiver computational cost in our first protocol. Our second protocol has a much better (quasi-optimal) computational cost, but due to the somewhat more sophisticated algorithms which make use of both FHE and LHE cryptosystems in a delicate interaction, developing a practical implementation of the second protocol remains as future work.

References

1. BLOOM, B. H. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (1970), 422–426.
2. BOSTAN, A., LECERF, G., AND SCHOST, É. Tellegen’s principle into practice. In *Symbolic and Algebraic Computation, International Symposium ISSAC 2003, Drexel University, Philadelphia, Pennsylvania, USA, August 3-6, 2003, Proceedings* (2003), J. R. Sendra, Ed., ACM, pp. 37–44.
3. BRAKERSKI, Z., GENTRY, C., AND VAIKUNTANATHAN, V. (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory* 6, 3 (2014), 13:1–13:36.
4. BRICKELL, J., AND SHMATIKOV, V. Privacy-preserving graph algorithms in the semi-honest model. In *Advances in Cryptology - ASIACRYPT 2005* (Berlin, Heidelberg, 2005), B. Roy, Ed., Springer Berlin Heidelberg, pp. 236–252.
5. CHEN, H., HUANG, Z., LAINE, K., AND RINDAL, P. 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 (2018), D. Lie, M. Mannan, M. Backes, and X. Wang, Eds., ACM, pp. 1223–1237.
6. CHOU, T., AND ORLANDI, C. 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* (2015), K. E. Lauter and F. Rodríguez-Henríquez, Eds., vol. 9230 of *Lecture Notes in Computer Science*, Springer, pp. 40–58.
 7. DAVIDSON, A., AND CID, C. 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* (2017), J. Pieprzyk and S. Suriadi, Eds., vol. 10343 of *Lecture Notes in Computer Science*, Springer, pp. 261–278.
 8. DEVROYE, L., AND MORIN, P. Cuckoo hashing: Further analysis. *Inf. Process. Lett.* 86, 4 (2003), 215–219.
 9. FRIKKEN, K. B. Privacy-preserving set union. In *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings* (2007), J. Katz and M. Yung, Eds., vol. 4521 of *Lecture Notes in Computer Science*, Springer, pp. 237–252.
 10. GARIMELLA, G., MOHASSEL, P., ROSULEK, M., SADEGHIAN, S., AND SINGH, J. 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* (2021), J. A. Garay, Ed., vol. 12711 of *Lecture Notes in Computer Science*, Springer, pp. 591–617.
 11. GARIMELLA, G., PINKAS, B., ROSULEK, M., TRIEU, N., AND YANAI, A. 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* (2021), T. Malkin and C. Peikert, Eds., vol. 12826 of *Lecture Notes in Computer Science*, Springer, pp. 395–425.
 12. GOLDREICH, O. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001.
 13. HANROT, G., QUERCIA, M., AND ZIMMERMANN, P. The middle product algorithm I. *Appl. Algebra Eng. Commun. Comput.* 14, 6 (2004), 415–438.
 14. JIA, Y., SUN, S.-F., ZHOU, H.-S., DU, J., AND GU, D. Shuffle-based private set union: Faster and more secure. In *31st USENIX Security Symposium (USENIX Security 22)* (Boston, MA, Aug. 2022), USENIX Association, pp. 2947–2964.
 15. KISSNER, L., AND SONG, D. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005* (Berlin, Heidelberg, 2005), V. Shoup, Ed., Springer Berlin Heidelberg, pp. 241–257.
 16. KOLESNIKOV, V., ROSULEK, M., TRIEU, N., AND WANG, X. 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* (2019), S. D. Galbraith and S. Moriai, Eds., vol. 11922 of *Lecture Notes in Computer Science*, Springer, pp. 636–666.
 17. LINDELL, Y. How to simulate it - A tutorial on the simulation proof technique. In *Tutorials on the Foundations of Cryptography*, Y. Lindell, Ed. Springer International Publishing, 2017, pp. 277–346.

18. NAOR, M., AND PINKAS, B. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA* (2001), S. R. Kosaraju, Ed., ACM/SIAM, pp. 448–457.
19. PINKAS, B., SCHNEIDER, T., AND ZOHNER, M. Faster private set intersection based on OT extension. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014* (2014), K. Fu and J. Jung, Eds., USENIX Association, pp. 797–812.
20. RAMANATHAN, S., MIRKOVIC, J., AND YU, M. BLAG: improving the accuracy of blacklists. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020* (2020), The Internet Society.
21. TU, B., CHEN, Y., LIU, Q., AND ZHANG, C. 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* (2023), W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds., ACM, pp. 2959–2973.
22. VON ZUR GATHEN, J., AND GERHARD, J. *Modern Computer Algebra (3. ed.)*. Cambridge University Press, 2013.
23. YI, X., PAULET, R., AND BERTINO, E. *Homomorphic Encryption and Applications*. Springer Briefs in Computer Science. Springer, 2014.
24. ZHANG, C., CHEN, Y., LIU, W., PENG, L., HAO, M., WANG, A., AND WANG, X. Unbalanced private set union with reduced computation and communication. *IACR Cryptol. ePrint Arch.* (2024), 1340.
25. ZHANG, C., CHEN, Y., LIU, W., ZHANG, M., AND LIN, D. 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* (2023), J. A. Calandrino and C. Troncoso, Eds., USENIX Association, pp. 337–354.

A About the Compatibility FHE-LHE

In this paper, our Protocols 1 to 3 combine FHE and LHE schemes. But, depending on the instantiation of both schemes, it is not always straightforward to preserve the overall correctness: denoting \mathbb{M}_L and \mathbb{M}_F the respective cleartext spaces of the LHE and FHE schemes, we have presented our protocols when the message spaces are both equal to the same finite field, $\mathbb{M}_F = \mathbb{M}_L = \mathbb{F}_q$. In this case, there are no compatibility issues and this is always possible as an FHE is by definition also an LHE.

Another situation is when $\mathbb{M}_F = \mathbb{F}_q$ and $\mathbb{M}_L = \mathbb{Z}_N$, for $q^2 < N$: for instance this is the case if BGV is used for the FHE and Paillier cryptosystem is used for the LHE. We show next that our Protocols 1 and 3 can benefit from such an instantiation.

Indeed, more precisely, in BGV, $\mathbb{F}_q = \mathbb{F}_{p^k} \simeq \frac{\mathbb{F}_p[X]}{(T(X))}$ 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 can 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 $q^2 < N$, 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 $\Psi(0) = 0$ but Ψ does not preserve the arithmetic. In Protocols 1 and 3, the receiver sends $\tilde{\alpha}$, an FHE encryption of α , and $\hat{\beta}$, an LHE encryption of $\Psi(\beta)$, to the sender. The sender decrypts $\tilde{\alpha}$, and computes and sends in fact $\hat{e} \leftarrow \hat{\alpha} -_L \hat{\beta}$ as well as $\hat{\eta} \leftarrow \Psi(y) \times_L \hat{e}$ and $\hat{\nu} \leftarrow \Psi(y) \times_L (\hat{0} -_L \hat{e})$ (where $\hat{\alpha}$ is an encryption of $\Psi(\alpha)$). Therefore, after decryption, if $\alpha = \beta$, all three of e, η and ν are 0, as expected. The issue is when $\alpha \neq \beta$. There are two cases, depending whether a modular reduction was needed in e when subtracting homomorphically $\Psi(\alpha) \in [0, q)$ to $\Psi(\beta) \in [0, q)$:

- If $e \in [0, q)$, then $\eta \in [0, q^2)$ and just dividing η by e over the rationals recovers $\Psi(y) = \eta e^{-1} = \Psi(y) e e^{-1} \in \mathbb{Q}$.
- If now $e \in [N - q, N)$, then in fact $\nu = \Psi(y)(N - e) \in [0, q^2)$ and we need to divide over the rationals by $N - e$ instead to recover $\Psi(y) = \Psi(y)(N - e)(N - e)^{-1} \in \mathbb{Q}$.

The actual case is easily decided in view of the deciphering of \hat{e} . Overall, we have shown that even in this BGV/Paillier-like situation, the receiver can compute back $y = \Psi^{-1}(\Psi(y)) \in \mathbb{F}_q$ if and only if $\alpha \neq \beta$, that is if and only if $P(y) \neq 0$.

For some other situations, an oblivious transfer [18] could be used to end the protocol instead, at the cost of an increase of the number of rounds.

B Security Proofs

We simultaneously do the simulation proof for Protocols 1 to 3, as their constructions are close. We also explicit the proof in the situation where the plaintext spaces for FHE and LHE are the same finite field $\mathbb{M} := \mathbb{F}_q$ (in the BGV/Paillier-like situation, otherwise, the analysis of Appendix A shows that the security is preserved).

We assume that both the FHE and LHE schemes used are IND-CPA secure. In the following, the receiver \mathcal{R} and the sender \mathcal{S} are respectively called party 1 and 2, while the UPSU protocol is denoted Π . The 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 $f : \mathcal{P}(\mathbb{M}) \times \mathcal{P}(\mathbb{M}) \rightarrow \mathcal{P}(\mathbb{M}) \times \mathcal{P}(\mathbb{M})$ where \mathcal{P} denotes the power set and \mathbb{M}_F is the FHE cleartext space.

The ideal output-pair is $f(\mathbf{X}, \mathbf{Y}) = (f_1(\mathbf{X}, \mathbf{Y}), f_2(\mathbf{X}, \mathbf{Y})) = ((\mathbf{X} \cup \mathbf{Y}), \emptyset)$. As the protocol is parametrized with (upper bounds on) the set sizes $|\mathbf{X}|$ and $|\mathbf{Y}|$, those are implicitly revealed to both parties. 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_F and a ciphertext $\mathbf{F.E}_{pk_F}(R)$, we omit $\mathbf{F.E}_{pk}(R)$ in the view, because if we can simulate both R and pk_F , it is trivial to simulate $\mathbf{F.E}_{pk}(R)$. As we are proving the security of three protocols, we will

use the notation $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ to describe the view, where a is related to Protocol 1, b to Protocol 2 and c to Protocol 3.

In the following, we consider that the **Setup** algorithm has already been completed: it is just an exchange of two public keys pk_L and pk_F , respectively for the LHE and FHE scheme. Now, the views and the outputs of each parties are:

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

$$C_1 = \left\{ pk_L, sk_L, pk_F, \begin{bmatrix} \{k_i\}_{i \in [m]} \\ M \\ \{k_i\}_{i \in [m]} \end{bmatrix}, \begin{bmatrix} \tilde{\mathbf{e}} \\ \tilde{R} \\ \{\tilde{e}_i\}_{i \in [m]} \end{bmatrix}, \{e_{\pi(i)}, \eta_{\pi(i)}\}_{i \in [m]} \right\},$$

where $\begin{bmatrix} \tilde{\mathbf{e}} \\ \tilde{R} \\ \{\tilde{e}_i\}_{i \in [m]} \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{F.BSMEv}(P_{\mathcal{R}}, \tilde{\mathbf{y}}) \\ P_{\mathcal{R}} \bmod_{\tilde{P}_{\mathcal{S}}} \\ \mathbf{F.MEv}(P_{\mathcal{R}}, \{\tilde{y}_i\}_{i \in [m]}) \end{bmatrix}$, and for all $j \in [m]$, $e_j \leftarrow \mathbf{L.D}_{sk_L}(\tilde{e}_j)$, $\eta_j \leftarrow \mathbf{L.D}_{sk_L}(\tilde{\eta}_j)$. The content of the first message is:

$$M_1 = \left\{ \begin{bmatrix} \tilde{\mathbf{y}} \\ \tilde{P}_{\mathcal{S}}, \tilde{U}_l \\ \{\tilde{y}_i\}_{i \in [m]}, \tilde{U}_l \end{bmatrix} \right\},$$

while the content of the second message is

$$M_3 = \left\{ \{\widehat{e}_{\pi(i)}, \widehat{\eta}_{\pi(i)}\}_{i \in [m]} \right\},$$

for $\widehat{e}_{\pi(i)} \leftarrow \begin{bmatrix} \mathbf{L.E}_{pk_L}(P_{\mathcal{R}}(y_i)) \\ \mathbf{L.E}_{pk_L}((P_{\mathcal{R}} \bmod P_{\mathcal{S}})(y_i)) \\ \mathbf{L.E}_{pk_L}(P_{\mathcal{R}}(y_i)) \end{bmatrix}$ and $\widehat{\eta}_{\pi(i)} \leftarrow y_i \times_L \widehat{e}_{\pi(i)}$.

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

$$C_2 = \left\{ pk_F, sk_F, pk_L, \begin{bmatrix} \langle h_i \rangle_{i \in [m]} \\ \{H(y_i)\}_{i \in [m]}, \{\widehat{e}_i\}_{i \in [m]}, \pi \\ \{h_i\}_{i \in [m]} \end{bmatrix} \right\},$$

$$M_2 = \left\{ \begin{bmatrix} \tilde{\mathbf{h}} \\ \tilde{H} \\ \{\tilde{h}_i\}_{i \in [m]} \end{bmatrix}, \begin{bmatrix} \{\widehat{k}_i\}_{i \in [m]} \\ \widehat{M} \\ \{\widehat{k}_i\}_{i \in [m]} \end{bmatrix} \right\}.$$

– $\mathbf{output}_1^H(\mathbf{X}, \mathbf{Y}) = (\mathbf{X} \cup \{y_i \in \mathbf{Y} | P_{\mathcal{R}}(y_i) \neq 0\})$
 – $\mathbf{output}_2^H(\mathbf{X}, \mathbf{Y}) = \emptyset$

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

$$S_2(\mathbf{Y}, \emptyset) = \left(\mathbf{Y}, \left\{ pk_F, sk_F, pk_L, \left[\begin{array}{l} \langle r_i \rangle_{i \in [m]} \\ \{R(y_i)\}_{i \in [m]} \\ \{r_i\}_{i \in [m]} \end{array} \right], \left[\begin{array}{l} \{\widehat{r}_i - L \widehat{r}'_i\}_{i \in [m]} \\ \{\widehat{R}(y_i) - L \widehat{R}'(y_i)\}_{i \in [m]} \\ \{\widehat{r}_i - L \widehat{r}'_i\}_{i \in [m]} \end{array} \right], \pi' \right\}, \right. \\ \left. \left\{ \left[\begin{array}{l} \mathbf{flood}(\mathbf{F.E}_{pk_F}(\langle r_i \rangle_{i \in [m]})) \\ \mathbf{F.E}_{pk_F}(R) \\ \{\mathbf{F.E}_{pk_F}(r_i)\}_{i \in [m]} \end{array} \right], \left[\begin{array}{l} \{\widehat{r}'_i\}_{i \in [m]} \\ \widehat{R}' \\ \{\widehat{r}'_i\}_{i \in [m]} \end{array} \right] \right\} \right),$$

where $m = |\mathbf{Y}|$, pk_F, sk_F, pk_L are obtained from the **Setup** algorithm, $r_i, r'_i \xleftarrow{\$} \mathbb{M}$ for all $i \in [m]$, $R, R' \xleftarrow{\$} \mathbb{M}[X]$ of degrees $m - 1$, for $\widehat{r}_i, \widehat{r}'_i, \widehat{R}, \widehat{R}'$ their LHE encryptions with pk_L , and $\pi' \xleftarrow{\$} \mathfrak{S}_m$.

In the protocol, $\{h_i\}_{i \in [m]}$ (resp. H) are masked values and are thus indistinguishable from the random $\{r_i\}_{i \in [m]}$ (resp. R). Note that in Protocol 1, the $\{h_i\}_{i \in [m]}$ are sent encrypted under FHE, and \mathcal{S} has the decryption key. The algorithm **flood** then ensures privacy (and circuit privacy): both the simulated and the real ciphertexts have the same lowest possible size and their noise are flooded. The $\{k_i\}_{i \in [m]}$ (resp. M) and π are random values, thus easily simulated with random values $\{r'_i\}_{i \in [m]}$ (resp. R') and π' . By performing the operations from the protocol to the simulated values, we obtain, for every subsets $\mathbf{X}, \mathbf{Y} \subset \mathbb{M}$:

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

On the side of \mathcal{R} , a probabilistic polynomial-time algorithm S_1 taking as input the set \mathbf{X} and $(\mathbf{X} \cup \mathbf{Y})$ first computes a set $\mathbf{Z} = \{z_i\}_{i \in [m]} \subset \mathbb{M}$, with $m = |\mathbf{Y}|$, as following. 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 $(|\mathbf{X} \cup \mathbf{Y}| - |\mathbf{X}|)$ are the cleartexts in $\mathbf{X} \cup \mathbf{Y} \setminus \mathbf{X}$. The set \mathbf{Z} is indistinguishable from the set \mathbf{Y} . Indeed, the two sets have the same size, and $\mathbf{X} \cup \mathbf{Y} = \mathbf{X} \cup \mathbf{Z}$ by construction; in particular, $|\mathbf{X} \cap \mathbf{Y}| = |\mathbf{X} \cap \mathbf{Z}|$. All, the information given by the sender to the receiver on the set $\mathbf{X} \cap \mathbf{Y}$ are sent encrypted under FHE in the protocol, and the receiver does not have the decryption key. This implies, under the IND-CPA security of the scheme, that taking any subset of \mathbf{X} with size $|\mathbf{X} \cap \mathbf{Y}|$ can lead to an indistinguishable simulation. Then, as the sender's input set has been properly simulated, S_1 simulates $\mathbf{view}_1^H(\mathbf{X}, \mathbf{Y})$ this way:

$$S_1(\mathbf{X}, (\mathbf{X} \cup \mathbf{Y})) = \mathbf{view}_1^H(\mathbf{X}, \mathbf{Z})$$

Overall, we obtain for every subsets $\mathbf{X}, \mathbf{Y} \subset \mathbb{M}$:

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

C Efficient Homomorphic Algorithm Constructions

In the following algorithms, for a polynomial $A = \sum_{i=0}^d a_i Z^i$, we denote by $[A]_l^L := \sum_{i=l}^L a_i Z^{i-l}$, and $\overleftarrow{A} := \sum_{i=0}^d a_{d-i} Z^i$.

C.1 FHE Euclidean Remainder

We recall the Newton-iteration-based algorithm for polynomial Euclidean division. We present the fast version of [13]. 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, by reversing the polynomial coefficients, $\overleftarrow{A} = \overleftarrow{Q}\overleftarrow{B} + Z^{n-m+1}\overleftarrow{R}$, whence

$$\overleftarrow{Q} = \overleftarrow{A}\overleftarrow{B}^{-1} \pmod{Z^{n-m+1}}. \quad (5)$$

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

$$(\tilde{U}) = \begin{cases} \tilde{U}_0 = \tilde{1} \\ \tilde{U}_{k+1} = \tilde{U}_k \times_F \left(\tilde{1} -_F \left[\overleftarrow{\tilde{B}} \times_F \tilde{U}_k \right]_{2^k}^{2^{k+1}-1} Z^{2^k} \right) \end{cases} \pmod{Z^{2^{k+1}}} \quad (6)$$

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

$$\tilde{S} = A \times_F \tilde{U}_L \pmod{Z^{n-m+1}}, \text{ and} \quad (7)$$

$$\tilde{T} = \left[\overleftarrow{\tilde{B}} \times_F \tilde{U}_L \right]_{2^L}^{2^{L+1}-1} \times_F \left[\tilde{S} \right]_0^{n-m-2^L} \pmod{Z^{n-m+1-2^L}}. \quad (8)$$

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

$$\tilde{R} = A -_F \overleftarrow{\tilde{Q}} \times_F \tilde{B} \pmod{Z^m} \quad (9)$$

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 (\tilde{U}) requires two homomorphic products which means that computing \tilde{U}_L has depth $2L$. To obtain \tilde{R} , it requires 3 more products, so in total, this algorithm has a depth $2\lceil \log(n - m + 1) \rceil + 1$.

C.2 LHE multi-point evaluation

We adapt the algorithm presented in [2] to the LHE context. Let $\hat{H} = \sum_{i=0}^{m-1} \hat{h}_i Z^i$, $H \leftarrow \mathbf{L.D}_{sk}(\hat{H})$, and $y_1, \dots, y_m \in \mathbb{M}$. For the sake of clarity we assume that m is a power of two, but it is not mandatory in practice. The first step of the

algorithm consists in computing the following polynomials in clear, for $k = 0, \dots, \log m$ and $i = 1, \dots, 2^k$:

$$P_{\left(\frac{i}{2^k}\right)} := \prod_{j \in \left\{\frac{i-1}{2^k}m+1, \dots, \frac{i}{2^k}m\right\}} (Z - y_j) \quad (10)$$

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

$$B := \overleftarrow{P}_{\left(\frac{1}{1}\right)}^{-1} \pmod{Z^m}, \text{ and} \quad (11)$$

$$\hat{A} := \left[\overleftarrow{B} \times_L \hat{H} \right]_{m-1}^{2m-1}. \quad (12)$$

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

$$\hat{A}_{\left(\frac{i}{2^k}\right)} = \left[\overleftarrow{P}_{\left(\frac{i - (-1)^{(i \bmod 2)}}{2^k}\right)} \times_L \hat{A}_{\left(\frac{\lceil i/2 \rceil}{2^{k-1}}\right)} \right]_{\frac{m}{2^k}}^{\frac{m}{2^{k-1}}}. \quad (13)$$

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

C.3 FHE multi-point evaluation

Our goal is to evaluate a polynomial A of degree n in m evaluation points $\{y_1, \dots, y_m\}$ homomorphically. To ease the description of the algorithm, we will assume that $m = 2^l$. We will consider the polynomials $P_{\left(\frac{i}{2^k}\right)}$ presented in (10). If the $\{y_i\}_{i \in [m]}$ are encrypted in a FHE scheme, one can compute those polynomials homomorphically and obtain the following sequence (\tilde{P}) :

$$(\tilde{P}) = \begin{cases} \widetilde{P_{\left(\frac{i}{m}\right)}} = Z -_F \tilde{y}_i; & i \in [m] \\ \widetilde{P_{\left(\frac{i}{2^{k-1}}\right)}} = \widetilde{P_{\left(\frac{2i-1}{2^k}\right)}} \times_F \widetilde{P_{\left(\frac{2i}{2^k}\right)}}; & k \in [l], i \in [2^k] \end{cases} \quad (14)$$

The computation requires less than $\frac{1}{2}l\mathcal{M}_F(m) + \tilde{O}(m)$ arithmetic operations with a depth l . Through an adaptation of the Newton iterations, given the sequence (\tilde{P}) and the encrypted $\{y_i\}_{i \in [m]}$, one can compute the following sequence (\tilde{V}) ,

that gives encryption of the set $\left\{ \overleftarrow{P}_{\left(\frac{i}{2^{l-k}}\right)}^{-1} \bmod Z^{2^k} \right\}_{i \in [2^k]}$, for $k \in [0, l]$, as following:

$$(\tilde{V}) = \begin{cases} \widetilde{V}_0^{(i)} = \widetilde{1}; & i \in [m] \\ \widetilde{V}_1^{(i)} = \widetilde{1} +_F (\widetilde{y_{2i-1}} + \widetilde{y_{2i}})Z; & i \in [2^{l-1}] \\ \widetilde{V}_{k+1}^{(i)} = \widetilde{K}_0 - Z^{2^k} \left[\left[\widetilde{K}_0 \right]_0^{2^k-1} \times_F (\widetilde{K}_1 +_F \widetilde{K}_2) \right]_0^{2^k-1} \bmod Z^{2^{k+1}}; & i \in [2^{l-(k+1)}] \end{cases}$$

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

$$\begin{aligned} \widetilde{K}_0 &= \widetilde{V}_k^{(2i-1)} \widetilde{V}_k^{(2i)}, \\ \widetilde{K}_1 &= \left[\overleftarrow{P}_{\left(\frac{2i-1}{2^{l-k}}\right)} \times_F \widetilde{V}_k^{(2i-1)} \right]_{2^k}^{2^{k+1}-1}, \\ \widetilde{K}_2 &= \left[\overleftarrow{P}_{\left(\frac{2i}{2^{l-k}}\right)} \times_F \widetilde{V}_k^{(2i)} \right]_{2^k}^{2^{k+1}-1}. \end{aligned}$$

We remark that for all $k \in [0, l]$ and $i \in [2^{l-k}]$, $\widetilde{V}_k^{(i)}$ is an encryption of $\overleftarrow{P}_{\left(\frac{i}{2^{l-k}}\right)}^{-1} \bmod Z^{2^k}$. It requires less than $2(l-2)\mathcal{M}_F(m) + \widetilde{O}(m)$ arithmetic operations. Also, this algorithm has a depth $2(l-1)$. With another sequence of Newton iteration, we want to obtain homomorphically $\overleftarrow{P}_{\left(\frac{1}{1}\right)}^{-1} \bmod Z^{n-m+1}$ 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 (\tilde{U}) from (6):

$$(\tilde{U}) = \begin{cases} \tilde{U}_l = \widetilde{V}_l^{(1)} \\ \tilde{U}_{k+1} = \widetilde{U}_k \times_F \left(\widetilde{1} -_F \left[\overleftarrow{P}_{\left(\frac{1}{1}\right)} \times_F \widetilde{U}_k \right]_{2^k}^{2^{k+1}-1} Z^{2^k} \right) \bmod Z^{2^{k+1}} \end{cases}$$

As explained in Section C.1, 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 denote \widetilde{R}_l . We are now applying a multi-point evaluation algorithm different from the one presented in Section C.2, which consists in successive Euclidean remainders in the $P_{\left(\frac{i}{2^k}\right)}$. The previously computed $\overleftarrow{P}_{\left(\frac{i}{2^k}\right)}$ and $\widetilde{V}_k^{(i)}$ will help us to do this algorithm homomorphically through the following sequence.

For $k \in [0, l - 1]$:

$$(\widetilde{R}) = \begin{cases} \widetilde{R}_l^{(1)} = \widetilde{R}_l (= A \bmod_F \widetilde{P}_{(\frac{1}{l})}); \\ \widetilde{R}_k^{(2^{i-1})} = \widetilde{R}_{k+1}^{(i)} -_F \widetilde{P}_{(\frac{2^{i-1}}{2^{l-k}})} \left[\overleftarrow{V_k^{(2^{i-1})} R_{k+1}^{(i)}} \right]_{0}^{2^k-1} \bmod Z^{2^k}; i \in [2^{l-k}] \\ \widetilde{R}_k^{(2^i)} = \widetilde{R}_{k+1}^{(i)} -_F \widetilde{P}_{(\frac{2^i}{2^{l-k}})} \left[\overleftarrow{V_k^{(2^i)} R_{k+1}^{(i)}} \right]_{0}^{2^k-1} \bmod Z^{2^k}; i \in [2^{l-k}] \end{cases} \quad (15)$$

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

$$\mathbf{F.D}_{sk_F} \left(\widetilde{R}_{k+1}^{(i)} \right) \bmod P_{(\frac{2^{i-1}}{2^{l-k}})} = \mathbf{F.D}_{sk_F} \left(\widetilde{R}_k^{(2^{i-1})} \right), \quad (16)$$

$$\mathbf{F.D}_{sk_F} \left(\widetilde{R}_{k+1}^{(i)} \right) \bmod P_{(\frac{2^i}{2^{l-k}})} = \mathbf{F.D}_{sk_F} \left(\widetilde{R}_k^{(2^i)} \right). \quad (17)$$

In particular, we have :

$$\mathbf{F.D}_{sk_F} \left(\widetilde{R}_0^{(i)} \right) = A(y_i) \quad (18)$$

Finally, computing homomorphically all the $\left\{ \widetilde{R}_0^{(i)} \right\}_{i \in [m]}$, given \widetilde{R}_l , all the $\widetilde{P}_{(\frac{i}{2^k})}$

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.