Privacy-Preserving Multi-Party Search via Homomorphic Encryption with Constant Multiplicative Depth

Mihail-Iulian Pleşa[1,2,3][0000−0001−5954−7199] and Ruxandra F. Olimid[1,2][0000−0003−3563−9851]

[1] Department of Computer Science, University of Bucharest, Romania
[2] Research Institute of the University of Bucharest (ICUB), Romania
[3] Orange Services, Bucharest, Romania
mihail.plesa@orange.com, ruxandra.olimid@fmi.unibuc.ro

Abstract. We propose a privacy-preserving multiparty search protocol using threshold-level homomorphic encryption, which we prove correct and secure to honest but curious adversaries. Unlike existing approaches, our protocol maintains a constant circuit depth. This feature enhances its suitability for practical applications involving dynamic underlying databases.

Keywords: Leveled Homomorphic Encryption (LHE) · Multi-Party Computation (MPC) · Privacy-preserving searching

1 Introduction

In today's data-driven world, vast amounts of information are generated and stored across numerous distributed databases. These data sets, often scattered across different organizations and geographical locations, hold immense potential for valuable insights and analyses. However, the decentralized nature of this data poses significant challenges for comprehensive analysis, as traditional methods require data aggregation, which can be impractical because of privacy concerns and regulatory constraints. Although last years' advances made cryptographic primitives such as Multi-Party Computation (MPC) and Homomorphic Encryption (HE) feasible and usable in many real-case scenarios, there is still room for improvement.

Some industry-oriented applications include securely searching for an IP address across multiple blocklists, sharing threat intelligence data to mitigate cyber threats, conducting collaborative vulnerability assessments, and coordinating incident response efforts. Central to these applications is privacy-preserving searching in a multi-party environment.

Contribution. We propose a protocol that allows one party to perform a search over multiple independent databases. The security of our construction is based on the threshold version of a Leveled Homomorphic Encryption (LHE) scheme [5,

9, 8, 13, 11]. The primary distinction of our proposal, compared to other similar solutions, is that our protocol employs a constant number of consecutive multiplications, maintaining a constant multiplicative depth regardless of the database size. This feature makes the protocol particularly well-suited for searches in dynamic data sets, where the number of entries is continually changing. Our contribution can be summarized as follows:

1. We introduce an algorithm with constant multiplicative depth for searching the 0 element into a linear array;
2. We use the proposed algorithm as a building block to construct a multi-party privacy-preserving searching protocol (based on LHE), which we prove secure in the honest-but-curious model;
3. We compare our protocol with other solutions in terms of the complexity of the searching algorithm, communication rounds, and multiplicative depth.

Outline. Section 2 presents the related works. Section 3 gives the necessary background, including the description of an LHE scheme. Section 4 defines the searching problem and discusses some general approaches to solving it. Section 5 introduces our protocol and gives the correctness and security proofs. Section 6 compares our proposal with existing protocols. The last section concludes and points out further directions of research.

2 Related works

Searching over encrypted data has, at its core, efficient algorithms for secure equality testing. Traditionally, these algorithms were based either on Homomorphic Encryption (HE) or on secure Multi-Party Computation (MPC) protocols [24, 12]. Regarding searchable encryption based on HE, in [25], the authors proposed a decentralized system for searching through genomic data using Fully Homomorphic Encryption (FHE). In [23], the authors describe a system for performing cloud-based computations of arbitrary depth over data under an FHE scheme. To address the challenge of noise growth in encrypted data, the authors utilized a specialized third-party equipped with hardware acceleration for FV encryption and decryption [13]. In [2], the authors proposed a method for searching over FHE-encrypted data realized by approximating the searching algorithm with a polynomial of degree logarithmic in the size of the database. Their method is based on the evaluation of the searching circuit over encrypted data within multiple distinct finite fields of small characteristics.

There are also other solutions to the problem of privacy-preserving collaborative analytics, based on garbled circuits and secret sharing [20, 6, 14, 22]. The main difference between these approaches and the HE approach lies in their handling of data and communication requirements. HE allows computations to be performed directly on encrypted data, maintaining continuous data confidentiality with typically fewer communication rounds. In contrast, garbled circuits and secret sharing require splitting the data into parts and distributing these parts

among multiple parties, which necessitates multiple rounds of communication and interaction to perform computations and reconstruct the results.

Over the years, several surveys on searchable encryption have been published [3, 7, 15, 4].

3 Preliminaries

3.1 Notations

Let $\mathbb{Z}^+$ be the set of positive integers, including 0. We represent uni-dimensional arrays by bold values, e.g., $\mathbf{v}$. We denote by $x \leftarrow^R X$ a uniformly random sampling of x from the set X. Let λ be the security parameter. For (pk, sk) a public-private key pair, let $[m]$ be the encryption of m (we ignore to index by pk for encryption and sk for decryption whenever this is clear from the context). Similarly, $[\mathbf{v}]$ represents a uni-dimensional array of ciphertexts.

Let $\mathcal{P} = \{P_1, \ldots, P_p\}$ be the set of p, $p \geq 2$, parties that take part in the searching protocol. Without losing generality (after a possible reordering), let P_1 be the querying party. Each party owns a private database $\mathbf{D}_i, i = 1, \ldots, p$, which we assume to be in the form of a vector of elements.

3.2 Homomorphic encryption

Overview. Homomorphic Encryption (HE) is an encryption scheme that supports homomorphic operations on encrypted data without performing decryption. The *multiplicative depth* of a circuit is the maximum number of multiplications along any path of the circuit and usually represents a bottleneck: a large multiplicative depth normally conducts to large parameters and ciphertexts, thus a high evaluation time.

A Fully HE (FHE) scheme allows any number of computations and the evaluation of arbitrary circuits of unbounded depth, while a Leveled HE (LHE) allows a predetermined number of operations and, thus, the evaluation of circuits of bounded depth. LHE schemes have a larger ciphertext expansion, but each operation is cheaper than in normal HE schemes [1].

Description. Let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an LHE scheme. Figure 1 defines the main homomorphic operations supported by an LHE scheme.

Besides the public encryption key pk and the private decryption key sk, these schemes employ a public evaluation key to facilitate homomorphic operations on encrypted data. Specifically, there is a public evaluation key evk for homomorphic multiplication [9, 8, 13, 11]. Also, note that $\mathsf{EvalSign}$ is not a native homomorphic operation, but we include it in Figure 1 because of its significance in our protocol. This functionality can be implemented using $\mathsf{EvalAdd}$ and $\mathsf{EvalMul}$ through polynomial representation [16, 17].

In this paper, we explore the use of BGV, BFV, and CKKS schemes, any of which can instantiate the proposed protocol [9, 8, 13, 11, 18], with CKKS being

Let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an LHE scheme, where:

- $\mathsf{KeyGen}(1^\lambda)$: on input the security parameter λ, it returns a pair (pk, sk) of public-private keys and an evaluation key evk;
- $\mathsf{Enc}_{\mathsf{pk}}(m) \rightarrow [m]$: on input the public key pk and a plaintext m, it returns a ciphertext $[m]$;
- $\mathsf{Dec}_{\mathsf{sk}}([m]) \rightarrow m'$: on input the private key sk and a ciphertext $[m]$, it returns a plaintext m' ($m' = m + e$ for some small enough, acceptable error e).

Then Π supports the evaluations (in the following, we ignore the errors and use the public key pk whenever no other key is explicitly mentioned):

1. $\mathsf{EvalAdd}([m_1], [m_2]) \rightarrow [m_1 + m_2]$: on input the ciphertexts $[m_1]$ and $[m_2]$ (the encryption of two plaintexts m_1 and m_2 respectively), it returns $[m_1 + m_2]$ a valid encryption of $m_1 + m_2$;
2. $\mathsf{EvalMul}_{\mathsf{evk}}([m_1], [m_2]) \rightarrow [m_1 m_2]$: on input the ciphertexts $[m_1]$ and $[m_2]$ (the encryption of two plaintexts m_1 and m_2, respectively), it returns a ciphertext $[m_1 m_2]$ that encrypts $m_1 m_2$;
3. $\mathsf{EvalAddScalar}([m], \alpha) \rightarrow [m + \alpha]$: on input the ciphertext $[m]$ (the encryption of a plaintext m) and the scalar α, it returns a ciphertext $[m+\alpha]$ that encrypts $m + \alpha$;
4. $\mathsf{EvalMulScalar}([m], \alpha) \rightarrow [\alpha m]$: on input the ciphertexts $[m]$ and the scalar α, it returns a ciphertext $[\alpha m]$ that encrypts αm;
5. $\mathsf{EvalSign}([m]) \rightarrow [b]$: on input a ciphertext $[m]$ (the encryption of a plaintext m), it returns [b], the encryption of a bit b indicating the sign of m: $b = 0$, for $m \leq 0$ and $b = 1$, otherwise.

Fig. 1. Leveled HE - Simplified description

distinct in its design to handle approximate data. The security of such schemes is based on the Ring Learning With Errors problem (RLWE) [21]. Both BGV and BFV are proven to be IND-CPA secure [9, 8, 13]. CKKS [11], with the modifications in [18], has been proven to be IND-CPA-D secure, an IND-CPA variant specific for approximate encryption [19][4].

Threshold LHE. Our protocol makes use of the Threshold FHE (TFHE) introduced in [5]. In this context, the parties $\{P_1, \ldots, P_p\}$ in $\mathcal{P}$ collaborate to create a unique public encryption key pk, with each party receiving a distinct share sk_i of the corresponding private key sk. Thus, the key generation algorithm $\mathsf{TFHE.KeyGen}(setup)$ becomes a multi-party protocol in which every party

[4] IND-CPA-D extends IND-CPA to capture resistance against passive attacks for approximate HE schemes by allowing access to a decryption oracle (in approximate schemes the adversary does not have, by construction, the ability to decrypt even if he/she knows the public key and the homomorphic computation being performed). IND-CPA-D is equivalent to IND-CPA for exact HE schemes.

$P_i, i = 1, \ldots, p$ outputs the public key pk, the public evaluation key evk, and a secret private-key share sk_i. The encryption and evaluation algorithms remain the same as for the underlying FHE. The decryption algorithm also becomes a multi-party protocol $\mathsf{TFHE.Dec}_{sk_1,\ldots,sk_p}([m])$. Hence, $\mathsf{TFHE.Dec}$ enables the decryption of a ciphertext $[m]$ only when the parties in $\mathcal{P}$ allow decryption[5]. Note that $\mathsf{TFHE.Dec}$ is asymmetric in the sense that at the end of the protocol, only one party - the party that initiated the protocol - finds the plaintext (the plaintext remains hidden for the other parties). TFHE is proven secure in the Universal Composability (UC) framework [5].

4 The problem

4.1 Problem definition

Overview. A set of $p \geq 3$ parties $\mathcal{P} = \{P_1, \ldots, P_p\}$ posses each a local database $\mathbf{D}_1, \ldots, \mathbf{D}_p$. One of these parties, called the *querying party* (let this party be P_1, without loss of generality after a possible reordering), wishes to search for a particular element within the collection of all the other parties' databases. The problem asks for the search to be *privacy-preserving*, in the sense that the queried element and the query result should remain hidden to all except the querying party, and no other information (e.g., wrt the content of the databases) must be disclosed. In particular, in case of a positive search result, the querying party should not be able to identify the party (parties) that possess the queried element.

We assume an *honest-but-curious model*, where each party follows the protocol correctly but is interested in learning as much as possible. In this model, parties do not deviate from the protocol's prescribed description; however, they may attempt to infer additional information from the data they observe during the execution of the protocol.

Privacy requirements. The privacy requirements are as follows, and they should hold, at least up to a negligible probability:

1. The querying party P_1 should learn nothing about the individual databases $\mathbf{D}_2, \ldots, \mathbf{D}_p$ apart from the fact that the searched element exists or not in at least one database (except its own $\mathbf{D}_1$).
2. Any database owner except the querying party P_1 should learn nothing about the searched element, the result of the protocol, or the individual databases.
3. Any coalition of parties that excludes the querying party P_1 should learn nothing about the searched element, the result of the protocol, or the individual databases.

The third privacy requirement generalizes the second requirement, extending it from single parties to coalitions of parties, which only strengthens the protocol's security.

[5] In fact, this is an all-or-nothing (i.e., the threshold equals the total number of parties) decryption because all p parties need to cooperate.

Operational requirements. We assume a dynamic setup in which participants can join or leave the group at any time. Additionally, the local databases are not static; they can undergo changes such as updates, deletions, and additions of elements. Considering the dynamic nature of the group of participants, we assume an appropriate key management system, which is beyond the scope of this paper. For instance, when the group of parties is altered, the keys are regenerated to ensure continued security and privacy.

4.2 Generic approaches to solve the problem

We discuss different approaches to solve the problem. We do not instantiate these solutions but keep them in terms of generic cryptographic building blocks.

Multi-Party Computation (MPC). The problem reduces to a secure MPC protocol where each party inputs its array, and the querying party inputs the value to be searched for. The output is 1 if there exists at least one party other than the querying party that has the value of interest and 0 otherwise. A similar approach can use the appurtenance of a set: all parties, except the querying party, compute a joined (encrypted) database using MPC, and the querying party further checks the appurtenance of the queried element to this set.

Private Set Intersection (PSI). The querying party inputs to a PSI protocol a set containing the searched element only. To maintain the privacy of the searching party, one might consider masking it within a vector populated with randomly chosen elements that are guaranteed not to belong to the database, such as by selecting elements outside a predefined range for the database entries. By construction, the PSI protocol reveals only the intersection of the sets and nothing else. This approach requires multi-party *unbalanced* PSI, a primitive that still raises many open questions.

Simplified select query. More complex solutions based on garbled circuits and secret sharing can also be employed for privacy-preserving general queries [20, 6, 14, 22]. Within this context, our problem can be formulated as a straightforward membership query.

Our method. Our approach is modeled as a privacy-preserving searching problem based on leveled HE that requires a constant number of communication rounds regardless of the number of parties or the dimension of their databases. A comparison (e.g., in terms of performance evaluation) of our protocol with the approaches mentioned above (e.g., a PSI-based solution) is left for future work.

5 Our protocol

5.1 0-searching algorithm

We introduce Algorithm 1 - 0SA for searching the 0 element within an arbitrary array of length n with non-negative entries, $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, $v_i \geq 0$. The

search algorithm receives as input the array $\mathbf{v}$ and outputs s, which is negative or zero if the $\mathbf{v}$ contains at least one 0 and is strictly positive otherwise. We define a function $H : \mathbb{Z}^+ \rightarrow \{0, 1\}$ as follows:

$$H(x) = \begin{cases} 1, \text{if } x > 0, \\ 0, \text{otherwise.} \end{cases} \tag{1}$$

Correctness (0SA). We consider the two possible cases: (a) $\mathbf{v}$ contains no 0 element, and (b) $\mathbf{v}$ contains at least one 0 element. (a) Suppose that the array $\mathbf{v}$ contains no 0 elements (i.e., $\forall v_i \in \mathbf{v}, v_i \neq 0$). Then, $s = n(1 - r)$. Since $r < 1$, it follows that $s > 0$. (b) Suppose that the array $\mathbf{v}$ contains k elements equal to 0, $1 \leq k \leq n$. Then, $s = (n - k) - nr$. Since $r \geq \frac{n-1}{n} \geq \frac{n-k}{n}$, it follows that $s \leq 0$.

The main advantage of the 0SA is that its circuit representation has a constant multiplicative depth, given by the number of consecutive multiplications used by the circuit that computes the function H. Since the number of consecutive multiplication is constant, the algorithm can be adapted to run over encrypted data using an LHE scheme. Algorithm 2 - 0EncSA delineates the procedure for searching over encrypted data by simply replacing the operations with the corresponding operations over ciphertexts (i.e., replaces sum with EvalAddScalar and H with EvalSign) The algorithm takes as input an array of ciphertexts and produces an encrypted result corresponding to the output of Algorithm 1 - 0SA if it had been executed directly on the corresponding plaintext.

Algorithm 1 0-Searching algorithm (0SA)

Input: $\mathbf{v}$
Output: s
$r \leftarrow^R \left[\frac{n-1}{n}, 1\right)$
$s = \sum_{i=1}^{n} (H(v_i) - r)$.
return s

Algorithm 2 0-Searching algorithm over encrypted data (0EncSA)

Input: $[\mathbf{v}]$
Output: $[s]$
$r \leftarrow^R \left[\frac{n-1}{n}, 1\right)$
$[s] = \mathsf{EvalSign}([v_1])$
$[s] = \mathsf{EvalAddScalar}([s], -r)$
for $i = 2$ **to** n **do**
  $[aux] = \mathsf{EvalSign}([v_i])$
  $[aux] = \mathsf{EvalAddScalar}([aux], -r)$
  $[s] = \mathsf{EvalAdd}([s], [aux])$
end for
return $[s]$

Algorithm 3 Squared Euclidean distance over encrypted data (EuclDist)

Input: $[m_1], m_2$
Output: $\left[(m_1 - m_2)^2\right]$
$m_2 \leftarrow -m_2$
$[c] \leftarrow \mathsf{EvalAddScalar}\,([m_1], m_2)$
$[d] \leftarrow \mathsf{EvalMul}\,(c, c)$
return $[d]$

Algorithm 4 Homomorphic sum of elements of an encrypted vector (SumVec)

Input: $[m_1], [m_2], \ldots, [m_n]$
Output: $[m_1 + m_2 + \cdots + m_n]$
$[s] \leftarrow [m_1]$
for $i = 2$ **to** n **do**
    $[s] \leftarrow \mathsf{EvalAdd}\,([\mathsf{s}], [\mathsf{m_i}])$
end for
return $[s]$

Note that 0EncSA requires computation over reals. Although many popular HE schemes do not accept reals by construction, significant work has been done regarding the processing of real numbers over encryption, with [11] containing a nice discussion in this respect.

Note. The absence of negative elements simplifies the problem: if **v** is ordered, comparing against the lowest element suffices. However, in our protocol, this simplification does not apply. For each query, we compute the distance between the encrypted query element and each database element, then search for 0 within this array of distances. As the array changes with each query, the simplification is inapplicable.

5.2 Protocol description

We generalize the 0-searching algorithm and present a protocol that allows privacy-preserving decentralized searching of an arbitrary value in the settings given in Section 4.1. The protocol is based on the threshold version of an FHE scheme [5], as explained in Section 3.2. The protocol outputs 1 if there is any database containing the query element and 0 otherwise.

We further assume that each database is a linear array of reals.[6] The protocol involves computing the squared Euclidean distance between two elements, one of which (the queried element) is encrypted. This functionality can be implemented using the homomorphic operations allowed by the scheme as described in Algorithm 3 - EuclDist. Algorithm 4 - SumVec implements another requested functionality: it computes the sum of a linear array over encrypted data.

[6] We leave the translation from a database to a vector out of the scope of this paper.

1. **Setup and Key Generation.** The parties $\{P_1, \ldots, P_p\}$ in $\mathcal{P}$:
   (a) Run the common setup.
   (b) Run the $\mathsf{TFHE.KeyGen}(setup)$ to obtain a common public key pk, an evaluation key evk and p private keys sk_i, $i = 1, \ldots, p$.
2. **Encrypted query.** The querying party, P_1:
   (a) Runs $\mathsf{TFHE.Enc_{pk}(m)} \rightarrow [m]$ to encrypt the query m using the public key pk.
   (b) Sends the ciphertext $[m]$ to each other party
   $$P_1 \rightarrow P_j : [m], \forall j = 2, \ldots, p$$
3. **Search.** Each party, $P_j, j = 2, \ldots, p$:
   (a) Uses the Algorithm 3 to compute the squared Euclidean distance between the encrypted query $[m]$ and each plaintext element of the database $\mathbf{D}_j$
   $$[d_j^k] = \mathsf{EuclDist}([m], D_j^k), \forall D_j^k \in \mathbf{D}_j.$$
   (b) Runs the searching $\mathsf{0EncSA}$ over the vector of encrypted squared distances obtained in the previous step
   $$[d_j] = \mathsf{0EncSA}([\mathbf{d}_j]), \text{ where } [\mathbf{d}_j] = ([d_j^k])_k.$$
   (c) Sends the encrypted result $[d_j]$ back to the querying party P_1
   $$P_j \rightarrow P_1 : [d_j]$$
4. **Compute final result.** The querying party, P_1:
   (a) Runs $\mathsf{EvalSign}$ on each ciphertext received from the $p-1$ parties, and then homomorphically sums the resulting ciphertexts
   $$[b_j] = \mathsf{EvalSign}\,([d_j]),\ 2 \leq j \leq p$$
   $$[s] = \mathsf{SumVec}\,([b_2], [b_3], \ldots, [b_p])$$
   (b) Involves in the threshold decryption protocol along the other parties $P_2, \ldots, P_p$ and decrypts the ciphertext $[s]$
   $$s = \mathsf{TFHE.Dec}_{sk_1,\ldots,sk_p}([s])$$
   (c) If the result value s equals $p-1$ then output 0; otherwise, output 1
   $$\mathsf{Output}\ 1 - [s == p-1]$$

Fig. 2. Our protocol on decentralized privacy-preserving searching

Figure 2 describes the protocol. The intuition of the protocol is as follows. Given a database in the form of a linear array and a query element, each party first computes the distance between the encryption of the queried element and each element from the array. The outcome of this step is another array with non-negative elements, which encrypts 0 strictly at every position where the corresponding element from the database is equal to the query. The problem is reduced, therefore, to determining if the encryption of 0 belongs to a linear array with non-negative numbers. Our $\mathsf{0EncSA}$ algorithm in Section 5.1 solves this problem without using multiplication operations.

Note that at step 4(b), we use the p-out-of-p threshold version of the underlying leveled HE scheme, so the decryption implies the cooperation of all parties, including the querying party P_1.

Theorem 1 (Correctness). *The protocol described in Figure 2 outputs* 1 *if there is at least one database with an entry equal to the query element and* 0 *otherwise.*

Proof. By computing the EuclDist in Step 3.(a), a database owner P_j, $2 \leq j \leq p$ obtains an array with encrypted non-negative entries. The array has the encryption of 0 in each position where the Euclidian distance is 0 (i.e., the element in the database is equal to the querying element) and the encryption of a positive number elsewhere. Thus, the problem of searching through the database is reduced to the problem of searching for the encryption of 0, which is solvable by 0EncSA. By construction, 0EncSA returns the encryption of a strictly positive number if and only if the encryption of 0 is not within the elements of the encrypted database. Hence, in step 3.(c) each party P_j, $2 \leq j \leq p$ returns to P_1 the encryption of a single element d_j, with $d_j \leq 0$ if and only if $\mathbf{D}_j$ contains the queried element m. After evaluating EvalSign function over all received encrypted results in step 4.(a), the querying party P_1 obtains an encrypted vector $[\mathbf{b}]$ that contains, on each position $j, j = 2, \ldots, p$ the encryption of $b_j = 0$, if m is in $\mathbf{D}_j$ and $b_j = 1$, if m is not in $\mathbf{D}_j$. Thus, $[s]$ in step 4.(b) is the encryption of $s = p-1$ if and only if no database $\mathbf{D}_2, \ldots, \mathbf{D}_p$ contains the querying element m. $[s]$ decrypts to s in step 4.(b). Finally, in step 4.(c), P_1 outputs 0 if no database $\mathbf{D}_2, \ldots, \mathbf{D}_p$ contains the querying element (i.e., s equals $p-1$) and 1 otherwise. □

It is important to note that the protocol accommodates both insertions and deletions for each database $\mathbf{D_i}, i = 1 \ldots p$. This flexibility ensures that the protocol can handle dynamic data sets without compromising the security of the overall system. This makes the protocol adaptable to various real-world applications where data is frequently changing.

5.3 Security Proof

Adversarial Model. To prove the security of our protocol, we use the Universal Composability (UC) framework that allows the composition of distinct protocols [10]. The framework delineates two types of executions: the *real* execution and the *ideal* execution. In the real execution, the parties interact according to the protocol in the presence of an adversary $\mathcal{A}$ and the environment $\mathcal{Z}$. Conversely, in the ideal execution, the parties interact with an ideal functionality, denoted as $\mathcal{F}_{\text{search}}$, which implements the protocol using a Trusted Third Party (TTP) in the presence of a simulated adversary $\mathcal{S}$ and the environment $\mathcal{Z}$. A protocol is considered secure if, for every adversary $\mathcal{A}$ in the real execution, there exists a simulated adversary $\mathcal{S}$ in the ideal execution such that the environment cannot distinguish between the real and ideal executions. We assume that all the parties are honest-but-curious, meaning they adhere to the protocol specifications but are motivated to glean additional information beyond their prescribed outputs (as explained in Section 4.1).

Theorem 2 (Security). *The protocol in Figure 2 securely realizes* $\mathcal{F}_{search}$ *in the presence of honest-but-curious adversaries.*

Ideal functionality $\mathcal{F}_{\text{search}}$

1. Each party $P_i, i = 2, \ldots, p$ securely sends the local database $\mathbf{D}_i$ to the TTP.
2. The querying party P_1 securely sends the queried element m to the TTP.
3. The TTP performs the search over all the databases $\mathbf{D}_i, i = 2, \ldots, p$ and returns 1 if the queried element m is found and 0 otherwise.

Proof (Sketch). We prove that the environment $\mathcal{Z}$ cannot distinguish between the real and the ideal executions, given the messages exchanged by the parties, until the querying party runs the *threshold decryption algorithm* in step 4.(b).

In the real execution, the environment $\mathcal{Z}$ sees $2p - 2$ messages:

1. The querying party P_1 sends a ciphertext encrypting the searched element to each of the $p-1$ database owners. These ciphertexts can be simulated by $\mathcal{S}$ by $p-1$ ciphertexts of random values. Since the encryption scheme is IND-CPA secure, the environment cannot distinguish between the ciphertexts.
2. Each database owner sends the output of the Algorithm 2 - 0EncSA. These output ciphertexts can also be simulated by $p - 1$ encryption of random elements because the encryption scheme is IND-CPA secure.

Given the fact that the *threshold decryption algorithm* used in step 4.(b) is secure against honest-but-curious adversaries in the UC framework [5], we conclude that our protocol in Figure 2 securely realizes $\mathcal{F}_{search}$ in the presence of honest-but-curious adversaries.

6 Protocol evaluation

We compare our encrypted data searching algorithm, 0EncSA, detailed in Algorithm 2, with the most cited privacy-preserving search solutions based on homomorphic encryption (HE). Our comparison focuses on multiplicative depth, communication rounds, and the complexity of the search algorithm. Although our procedure is designed to search for the 0 element in a non-negative array, it can be easily extended to search for any arbitrary value, as shown in Figure 2. The reason we look into the 0EncSA and not the final proposed protocol is mainly because the other works do not give results in the multi-party settings. Nevertheless, we are aware of the fact that the comparison might not be fair in the sense that 0EncSA allows a particular search and does not accept, by itself, general queries.

Table 1 summarizes the comparison with the protocols introduced in [25, 23, 2]. Multiplicative depth affects the running time due to linearly increasing noise, necessitating costly bootstrapping operations. The number of communication rounds impacts the network overhead and decides whether the participants must be online. Finally, the complexity of the search algorithm influences the overall efficiency of the solution. The most efficient protocol in terms of the searching

Table 1. Comparison with related work

Ref.	Communication rounds	Multiplicative depth	Overall Complexity
[25]	$O(m)$	$O(m)$	$O(n)$
[23]	$O(1)$	$O(m)$	$O(n)$
[2]	$O(1)$	$O(log(log(m)))$	$O(log(n))$
Our protocol	$O(1)$	$O(1)$	$O(n)$

*n is the number of entries in the database; m is the length of the queried element

complexity is [2], while our approach demonstrates the highest efficiency in terms of multiplicative depth. From a practical perspective, we conclude that Akavia et al.'s protocol [2] is better suited for static large databases, while our proposal is preferable for scenarios involving relatively small but dynamic databases.

7 Conclusions and future work

We proposed a new multi-party privacy-preserving searching protocol that maintains a constant multiplicative depth. We proved it correct and secure under the honest-but-curious adversarial model. We compared our proposal with existing works, highlighted its advantages, and concluded that it is appropriate for dynamic databases.

We have also briefly referred to generic approaches to solve the secure search problem. A detailed analysis and comparison to other works is out of the current goal, mostly because of the lack of space, but it is considered for an extension of the paper. We thus postpone the implementation and the numeric performance evaluation of the proposed solution for future work. It is also of interest to look into the performance of the solution in real use cases such as, e.g., the IP address search against multiple blocklists, as exemplified in the introduction.

Besides outputting a boolean result (1 if the element is found in any of the party's databases and 0 otherwise), our protocol can also number the parties that have the queried element in their databases. This is easily reached by changing the protocol's output in the last step. Further investigation of this functionality in different applications could be interesting. Finally, in the current definition, the problem asks that the querying party P_1 be internal to the set of parties. Future work will look into solutions where the queries can also originate from external parties.

Acknowledgements. This work was supported by a grant of the Ministry of Research, Innovation and Digitalization, CNCS/CCCDI - UEFISCDI, project number ERANET-CHISTERA-IV-PATTERN, within PNCDI IV.

References

1. Acar, Abbas and Aksu, Hidayet and Uluagac, A. Selcuk and Conti, Mauro: A

Survey on Homomorphic Encryption Schemes: Theory and Implementation. ACM Comput. Surv. **51**(4) (jul 2018). https://doi.org/{10.1145/3214303}
2. Akavia, Adi and Feldman, Dan and Shaul, Hayim: Secure search on encrypted data via multi-ring sketch. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 985–1001 (2018)
3. Amorim, Ivone and Costa, Ivan: Leveraging Searchable Encryption through Homomorphic Encryption: A Comprehensive Analysis. Mathematics **11**(13) (2023). https://doi.org/{10.3390/math11132948}
4. Andola, N., Gahlot, R., Yadav, V.K., Venkatesan, S., Verma, S.: Searchable encryption on the cloud: a survey. The Journal of Supercomputing **78**(7), 9952–9984 (2022)
5. Asharov, Gilad and Jain, Abhishek and López-Alt, Adriana and Tromer, Eran and Vaikuntanathan, Vinod and Wichs, Daniel: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31. pp. 483–501. Springer (2012)
6. Bater, Johes and Elliott, Gregory and Eggen, Craig and Goel, Satyender and Kho, Abel and Rogers, Jennie: smcql: Secure Querying for Federated Databases. Proceedings of the VLDB Endowment **10**(6) (2017)
7. Bösch, C., Hartel, P., Jonker, W., Peter, A.: A survey of provably secure searchable encryption. ACM Computing Surveys (CSUR) **47**(2), 1–51 (2014)
8. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Annual cryptology conference. pp. 868–886. Springer (2012)
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
10. Canetti, Ran: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145. IEEE (2001)
11. Cheon, Jung Hee and Kim, Andrey and Kim, Miran and Song, Yongsoo: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. pp. 409–437. Springer (2017)
12. Couteau, Geoffroy: New protocols for secure equality test and comparison. In: International Conference on Applied Cryptography and Network Security. pp. 303–320. Springer (2018)
13. Fan, Junfeng and Vercauteren, Frederik: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
14. Han, Feng and Zhang, Lan and Feng, Hanwen and Liu, Weiran and Li, Xiangyang: Scape: Scalable collaborative analytics system on private database with malicious security. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE). pp. 1740–1753. IEEE (2022)
15. Handa, R., Krishna, C.R., Aggarwal, N.: Searchable encryption: a survey on privacy-preserving search schemes on encrypted outsourced data. Concurrency and Computation: Practice and Experience **31**(17), e5201 (2019)
16. Iliashenko, I., Zucca, V.: Faster homomorphic comparison operations for bgv and bfv. Proceedings on Privacy Enhancing Technologies **2021**(3), 246–264 (2021)
17. Lee, E., Lee, J.W., Kim, Y.S., No, J.S.: Optimization of homomorphic comparison algorithm on rns-ckks scheme. IEEE Access **10**, 26163–26176 (2022)

18. Li, B., Micciancio, D., Schultz, M., Sorrell, J.: Securing approximate homomorphic encryption using differential privacy. In: Annual International Cryptology Conference. pp. 560–589. Springer (2022)
19. Li, Baiyu and Micciancio, Daniele: On the security of homomorphic encryption on approximate numbers. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 648–677. Springer (2021)
20. Liagouris, John and Kalavri, Vasiliki and Faisal, Muhammad and Varia, Mayank: {SECRECY}: Secure collaborative analytics in untrusted clouds. In: 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23). pp. 1031–1056 (2023)
21. Lyubashevsky, Vadim and Peikert, Chris and Regev, Oded: On ideal lattices and learning with errors over rings. In: Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. pp. 1–23. Springer (2010)
22. Poddar, Rishabh and Kalra, Sukrit and Yanai, Avishay and Deng, Ryan and Popa, Raluca Ada and Hellerstein, Joseph M: Senate: a Maliciously-Secure MPC platform for collaborative analytics. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 2129–2146 (2021)
23. Roy, Sujoy Sinha and Vercauteren, Frederik and Vliegen, Jo and Verbauwhede, Ingrid: Hardware assisted fully homomorphic function evaluation and encrypted search. IEEE Transactions on Computers **66**(9), 1562–1572 (2017)
24. Saha, Tushar Kanti and Koshiba, Takeshi: Private equality test using ring-LWE somewhat homomorphic encryption. In: 2016 3rd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE). pp. 1–9. IEEE (2016)
25. Yamamoto, Yuri and Oguchi, Masato: A Decentralized System of Genome Secret Search Implemented with Fully Homomorphic Encryption. In: 2017 IEEE International Conference on Smart Computing (SMARTCOMP). vol. , pp. 1–6 (2017). https://doi.org/{10.1109/SMARTCOMP.2017.7946977}