Secure Search on Multi-key Homomorphically Encrypted Data with Finite Fields

Buvana Ganesh* and Paolo Palmieri

School of Computer Science & IT,
University College Cork, Ireland
b.ganesh@cs.ucc.ie, p.palmieri@cs.ucc.ie

Abstract. Homomorphic Encryption (HE) is a very attractive solution to ensure privacy when outsourcing confidential data to the cloud, as it enables computation on the data without decryption. As the next step, searching this homomorphic data becomes necessary to navigate it in the server. In this paper, we propose a novel algorithm to search homomorphically encrypted data outsourced to an untrusted server and shared with multiple users. We optimize the steps involved in the process to reduce the number of rounds of communication. We use an order-preserving encoding [21] to batch the data with multi-key HE cryptosystems [9] to reduce the multiplicative depth of the equality circuits and enable direct comparison. Further, we use LEAF [22] to retrieve indices securely, and SealPIR [4] to retrieve the values obliviously to the user. Overall, we provide an efficient end-to-end framework for searching shared data in a semi-honest server.

Keywords: Homomorphic encryption · Secure search · Encrypted databases · Data Sharing

1 Introduction

Outsourcing data and computation to the cloud has become extremely common today. However, many cloud users would like to be ensured that their data remains secure when utilized from the cloud. A way to achieve this, albeit at the cost of increased computation, is by using homomorphic encryption. Let's consider an example with an hospital that is outsourcing all its data to an untrusted Cloud Service Provider, and its staff are trying to retrieve patient records from the hospital data on a daily basis. One can also consider banks performing analytics on a subset of its customers. Recent advances in the state of the art make it possible to securely search and compute, even run AI algorithms, on this data when it is homomorphically encrypted.

However, encrypted search remains a computationally tedious task as it is based on comparisons on non-deterministic data. Currently, many methods are available to search for encrypted data. For a statistical operations on data, differential privacy is a viable solution. Secure multiparty computation, Private Set Intersection, searchable encryption, and encrypted database schemes are available based on the requirement. The standard database encryption and searchable schemes tend to be either computationally intensive or extremely leaky.

To solve this, Akavia et al. [1] were one of the first to provide viable schemes for homomorphic search with Fully Homomorphic Encryption (FHE) schemes. Secure outsourcing of computation using FHE involves the client sending the ciphertext $\mathbf{x}$ to the Cloud or the untrusted server and executing some function f on the data. The client receives the ciphertext $\mathbf{r}$ encrypting the output $\mathbf{r} = f(\mathbf{x})$. This gives a single round protocol with communication complexity proportional only to the sizes of the input and output ciphertexts. Typically, FHE is secure under the Chosen Plaintext attack and this semantic security ensures that the server learns nothing about the plaintext input and output from the ciphertexts. Some of these principles can be extended to search in general.

But if we are to extend this scenario to one where the data is shared with multiple parties, we find that we need more than one round of communication to search securely. In this paper, we explore searching data outsourced to the cloud with multiple keys. Further, once we can perform operations on the data collectively, we perform optimized search algorithms on the data and reduce them from linear to a logarithmic number of multiplications. We use the LEAF algorithm [22] and the SealPIR protocol [4] to illustrate the architecture.

* Buvana Ganesh is supported by a PhD scholarship funded by the Science Foundation Ireland Centre for Research Training in Artificial Intelligence under Grant No. 18/CRT/6223.

1.1 Contributions

In our work, we introduce an architecture to exploit multi-key homomorphic encryption schemes for Secure Search against semi-honest servers and malicious users. We leverage inherent aspects of the HE construction like batching in order to search better with comparison circuits using the vector of field elements encoding as it preserves order. Methods to enable efficient comparison on encrypted data are still explored in the literature. But we exploit the underlying finite fields in the HE cryptosystems to enable comparisons. Our scheme is modular and so all three major components can be substituted according to the application required. The architecture is agnostic to the HE cryptosystem as long as it supports batching. It also achieves better security in terms of access patterns, search patterns leakage, etc, compared to the other searchable encryption schemes because of the use of SealPIR for oblivious retrieval. Also, unlike searchable encryption schemes, further computations can be performed on the encrypted data because of HE. Overall, our paper is one of the first to explore secure search in a multi-key setting. We address different problems faced in creating the protocols and provide possible solutions to those problems.

1.2 Outline

In the following sections, we introduce Homomorphic Encryption (Sec. 2), and multi-key homomorphic encryption (Sec. 2.1) along with different methods to pack data in HE. Then we explore some feasible solutions for performing secure search (Sec. 3.1) and search protocols on HE algorithms (Sec. 3.2), in particular, that perform reasonably well. We then introduce our architecture and how to choose different components based on the security requirements. Then the security (Sec. 5) and the performance (Sec. 6) of the architecture are analyzed for our architecture. We provide some possible extensions of the work for the future and research questions (Sec. 7) at the end.

2 Homomorphic encryption

An encryption scheme is homomorphic (HE) if we can perform computations on encrypted data without decryption. Fully HE schemes can perform any number of operations on ciphertexts whereas Leveled HE schemes can perform limited operations to a level L to avoid decryption failure. The schemes that are fully homomorphic can perform any operation of any circuit size including arithmetic operations like addition, subtraction, and multiplication on encrypted data. Some of the HE libraries also allow exponentiation, square, signing, and therefore subtraction. In Levelled HE schemes, modulus switching is used to convert a noisy ciphertext under one modulus say Q, into an equivalent ciphertext modulo $q < Q$ and co-prime, such that both decrypts to the same plaintext because the noise is reduced by a factor of nearly Q/q. In FHE schemes, bootstrapping is used to reduce the noise in the ciphertext but this is too expensive.

Improved versions of the most popular schemes like BFV [13], BGV [8], CKKS [11] use SIMD to pack more data into a single ciphertext. Below, we illustrate an FHE scheme and its components using the BFV cryptosystem. The polynomial ring R is similar for other FHE schemes like BGV, CKKS, etc following the Ring Learning with Errors (RLWE) assumption. This class of schemes is called BGV-like cryptosystems because they work around similar assumptions and structures.

Let λ be the security parameter. For a levelled version without bootstrapping, at most L levels of evaluations are possible in the FHE scheme. The levelled scheme FV consists of the algorithms $SecretKeyGen$, $PublicKeyGen$, $EvalKeyGen$, $Encrypt$, $Decrypt$. Let t and q be the plaintext and ciphertext moduli, n a power of 2 be the degree of the cyclotomic polynomial. The base of the cryptosystem is the ring $R_i = Z_i[x]/(x^n) + 1$ for $i = t, q$. Let Q be the base for relinearization once the noise increases during multiplication. For $l = \lfloor log_w q \rfloor$, there are $l+1$ polynomials when the elements of R_q are split coefficient-wise. Let χ be the truncated discrete Gaussian error distribution with standard deviation σ.

– Sample $s \leftarrow R_2$, the secret key sk.

$$SecretKeyGen(1^\lambda, L) \leftarrow sk$$

– Sample a random vector $a \leftarrow R_q$ and the noise $e \leftarrow \chi$,

$$PublicKeyGen(sk) \leftarrow pk = (pk_0, pk_1) = ([(as + e)]_q, a)$$

– For $i \in \{0, ..., l\}$, sample random values $a_i \leftarrow R_q, e_i \leftarrow \chi$

$$EvalKeyGen(sk, Q) \leftarrow evk = ([(a_i s + e_i) + Q_i s^2]_q, a_i)$$

– Sample $u \leftarrow R_2$, and $e_1, e_2 \leftarrow \chi$. For the message $m \in R_t$ and pk.

$$Encrypt(pk, m) \leftarrow ct = ([\Delta m + pk_0 u + e_1]_q, [pk_1 u + e_2]_q)$$

– For $s = sk$, $c_0 = ct[0]$, and $c_1 = ct[1]$,

$$Decrypt(sk, ct) \leftarrow [\lfloor \frac{t}{q} [c_0 + c_1 s]_q \rceil_t]$$

2.1 Multikey Fully Homomorphic Encryption

Asharov et al. [5] introduced Threshold FHE schemes for multiple parties under a semi-malicious setting. Their scheme follows the Regev HE cryptosystem and uses distributed decryption. Following this work, there are many schemes that are the multi-key versions (MKHE) [[9], [10]] of the popular schemes like BFV, BGV, TFHE, etc. These rely on the security of the single-key versions of the cryptosystems using the LWE assumption. Aloufi et al. provide an effective insight into comparing the different multikey and threshold schemes in [3].

Primarily keys can be handled in two ways.

– In MKFHE schemes, a new user gets a key as they enter the computation. The evaluation key of the owner and the participants are used to convert and form shared ciphertexts that are extended.
– In threshold schemes, the public key is aggregated after the key generation. If a new user enters the system, the whole set-up phase has to be run again. But this enables computation between any party as they share a common public key.

The application plays a vital role in choosing which kind of MKHE is to be used. For certain applications with a trusted party in the communication, public and private keys are generated for everyone individually or by a trusted third party. In threshold-based schemes and the public key is aggregated. This stops more parties from entering the computation without running the setup again. On the other hand, a Common Reference String (CRS) with encryption of all the keys is distributed to enable separate decryption for any of the parties involved. In this case, any number of users are supported to enter the computation because it just involves generating a new pair of keys.

When all the secrets of the parties involved are required to decrypt a multi-key ciphertext, the decryption is *distributed* and not reliant on a trusted third party. This is common in threshold HE. Both Threshold HE and MKHE schemes are similar in requiring users to cooperate and run a distributed decryption protocol when retrieving the evaluated result.

Chen et al. [9] provide multi-key versions of BFV and CKKS with linear expansion of the ciphertext with the multiple keys. Also, we would like to keep the system dynamic by allowing parties to enter the computation with just KeyGen and not aggregation. We refer to this scheme as MKBFV and use it to illustrate the architecture. MKBFV uses CRS and a gadget vector in their scheme where all parties share a random polynomial vector The gadget vector and the corresponding function help with bit/base decomposition in relation to FHE, in this case, the linear expansion of the ciphertext. This helps in generating the trapdoors to enable identity-based operations.

MKBFV provides two ways to convert and expand the ciphertext, one where the key is converted followed by ciphertext relinearization, and one where the ciphertext is directly converted to the other keys using $\mathbf{Relin}(ct, (ev_{\mathcal{O}}, b_i))$ where $ct \in R_q^{k+1}$ is the ciphertext with k components for k users. They also provide bootstrapping methods and extend the scheme to an MKFHE scheme. We exploit these techniques to relinearize, enable partial decryptions and perform limitless computation on the data in our scheme.

2.2 Different Encoding methods

Traditional FHE schemes encoded and encrypted data bit by bit leading to a lot of expansion in the ciphertexts. Later, Smart and Vercauteren [20] introduced the Chinese Remainder Theorem-based batching to enable Single Instruction Multiple Data (SIMD) in FHE. This also speeds up arithmetic

operations in FHE. CRT-based packing is one of the best ways to pack a lot of data with less storage size. But the more data packed in a ciphertext, the harder it becomes to perform comparison operations on the encrypted data. Therefore, we have to find an efficient way of packing that does not reduce performance for arithmetic operations or comparison.

CRT-RNS packing Encoding the elements bit-wise is not very efficient and therefore schemes use Single instruction multiple data (SIMD) in order to speed up computation using the Chinese Remainder Theorem (CRT) which allows splitting a single big ring into smaller rings with certain constraints to allow Frobenius automorphism and Hensel lifting. The isomorphism splits the factors of the nth degree cyclotomic polynomial into irreducible polynomials of degree d using the CRT. We consider t to be prime and congruent to 1 mod $2n$ to pack n integers. As the CRT is performed in a finite field, the Residue Number System (RNS) is used and improved in various aspects. Frobenius maps take almost no multiplicative depth in FHE and are often used as substitutes for certain exponentiations. SIMD packing allows ordinary operations like addition, multiplication, scalar or otherwise, rotation within the slots. The special function IsNonZero uses the Frobenius automorphism. Rotation costs the same as relinearization if its particular evaluation keys are generated because of the precomputations. All these properties allow faster computation, especially arithmetic operations on encrypted data. But the CRT packing does not allow fast comparisons. One has to resort to F_2 for performing speedy comparison circuits.

Vector of Field Elements (VFE) One way to solve the comparison problem is by using an encoding that preserves order when packing ciphertexts. Any homomorphic encryption algorithm that uses ciphertext packing can be adapted to the VFE encoding [21] which packs the plaintext to preserve order. This is done by decomposing an integer in $Z \cap [0, q^{n.d})$ to a length-n vector of digits in $Z \cap [0, q^d)$, and further encoded to F_q^d. Here $Z \cap [0, q^{n.d})$ is isomorphic to $R_q^n = (Z_q[x]/f(x))^n$ where $f(x)$ is a polynomial of degree d. This implies a direct connection between schemes on polynomial rings with the finite fields.

It simply takes the base-q representation of the digits $x = \sum_{i=0}^{d-1} x_i q^i$ and maps it to the field element $\sum_{i=0}^{d-1} x_i t^i$, where t is the root of some degree d polynomial irreducible modulo q. For the parameters, $n.d = \lceil \Gamma / logp \rceil$. $l = \phi(m)/d'$ so we can pack $\lfloor l/n \rfloor$ elements in a single ciphertext.

The plaintexts in FHE schemes are usually represented using polynomials in finite rings which in turn correspond to vectors in $(F_{q^d})^l$ for $q \geq 2$. This is done by constructing isomorphisms with n-variate polynomials and further with higher dimensional fields to ease and enhance computation. With the split polynomial fields, the plaintext is in the form of length l vectors $\mathbf{v} = (v_1, ..., v_l)$ such that each $v_i \in F_{q^d}$. Let $[n]$ denote $\{0, 1, ..., n-1\}$ and $\mathbf{v} = (v_0, v_1, ...v_{d-1})$. The Field Elements methods connects fields of higher orders of p to a lower order higher dimensional field.

This space for VFE-encoded data is totally ordered and this enables comparison. Arithmetic operations like addition and multiplication are done component-wise. This type of packing also allows less expensive operations like Shift, Rotate, etc using the evaluation key. It also allows the Frobenius automorphism ($Frob : F_{q^d}^l \to F_{q^d}$ such that $\text{Frob}(evk, \mathbf{v}, i) = (v_1^{p^i}, ..., v_l^{p^i})$. Frobenius maps take every element to its dth power of the prime using the map which arises from a chain of automorphisms.

The isomorphisms between the plaintext spaces for Field Element encodings are given by

$$\text{FE - } \phi_1 : (F_q)^d \to F_{q^d}, \text{ where } \phi_1(\mathbf{v}) = \sum_{i=0}^{d-1} v_i t^i$$

$$\text{VFE - } \phi_2 : Z \cap [0, q^{n.d}) \to (F_{q^d})^n, \text{ where } \phi_2(a) = \mathtt{a} = (\phi_1(\mathbf{a_i}))_{i \in [n]}$$

2.3 Comparison circuits

Secure search methods using FHE often do not consider using the full capacity finite extension fields and their properties to reduce computation as F_p^d is the native space for the most widely used SIMD-capable schemes by BGV [8], and BFV [13], CKKS [11]. With the normal CRT based encoding, there are several works that perform comparisons done with the help of minimax polynomials or Lagrange interpolation of the circuit. These are approximate methods and are still expensive operations. Therefore we use the VFE methods to perform comparison circuits on finite fields [21].

The encrypted database and the query are matched together using several methods. A naive way is just by subtraction when the elements in the dataset are stored as an encrypted array. Else, if the

ciphertexts are polynomials, then each element is checked to be the root of a polynomial comprised of all queries as roots. This search polynomial method is easily expandable to conjunctive or disjunctive queries. We only consider the equality and less than circuits because other comparison circuits can be constructed from that.

Binary Equality Circuit for $x, y \in F_l$ are straightforward with multiplicative depth l. BUt this can be reduced using the properties of finite field extensions like order.

$$EQ_2(x, y) = \prod_{i=0}^{l-1} (1 - (x_i - y_i))$$

We can optimize packed Equality Circuits for $x, y \in F_{t^d}$ with multiplicative depth because of the repeated squaring using Fermat's little theorem.

$$EQ_{FE}(x, y) = (1 - (x - y)^{q^d - 1})$$

With the VFE encoding, comparison can be executed with only $logn + logd + 1$ multiplications in total. For the Less Than(LT) circuit evaluation on two inputs $x, y \in F_{p^d}$, process x and y with Decomp to get the decomposed d-tuple. Then evaluate $LT_{F_p}(x; y)$ and $EQ_{F_p}(x; y)$ where x and y $\in F_p^d$. Finally, combine the separate results according to Equation LT_{FE}, which is then used recursively over n components to calculate LT_{VFE}.

$$LT_{FE}(x; y) = LT_{F_p}(x_{d-1}; y_{d-1}) + \sum_{i=0}^{d-2} LT_{F_p}(x_i; y_i) \prod_{j=i+1}^{d-1} EQ_{F_p}(x_j; y_j)$$

3 Related Work

3.1 Search using Homomorphic encryption

Preserving the privacy of the data and enabling search is a delicate trade-off. Schemes like Searchable encryption, with or without ORAM, Private Information Retrieval by keywords, etc try to hide the *query content, search pattern, and access pattern* for the best security guarantees. Managing these in one or two rounds of communication with low bandwidth becomes a tedious task. Though there are schemes that provide these features as discussed in [14] , they come with high setup costs or refresh rates. Therefore, we consider schemes that only require homomorphic encryption. The schemes listed below suffer performance, security and functionality trade-offs but we observe that most functionalities are lost in order to accommodate security. There are many tools to perform secure search on data but we consider methods without false positives and those that retrieves all the data methodically using just Homomorphic encryption as the primitive.

Boneh et al. [6] presented Private Database Querying using homomorphic encryption in the two-party setting, there is only a client and a single database server. The client holds a secret key for an SWHE scheme, and the server has the corresponding public key and the database. Search-and-compute was one of the first to explore search in encrypted data using basic addition and multiplication to locate data.

Secure Pattern matching (SPM) is another method widely used to retrieve patterns, but the lookup time is almost that for the database. It only indicates if an element is the ith data element or if the element exists or not. Yasuda et al. [23] check for SPM using Hamming distance to measure equality.

Kim et al. [19] provide methods with unique minimal polynomials and correlated functions to polynomials to evaluate a function (specifically equality) with the minimum multiplicative depth by evaluating it in the form of the minimal polynomial expression. Their scheme also processes conjunctive and disjunctive queries. Kim et al. further extend their work in [18] to discuss wildcard matchings in semi-honest models.

Another method to retrieve not exact but relevant data from datasets is using kNN. Kim et al. [16] use kNN on HE to return only k queries and prevent volume-based attacks. They guarantee data privacy and query privacy but also conceal the data access pattern at the same time. A data group generated based on the Hilbert-curve order is considered to be a query processing unit. The search protocol uses k dimensional trees and encrypted indices for search and consists of 5 protocols with SMC and one Secure Bit Not protocol using secure squared Euclidean distance.

Bonte et al. [7] explore packed ciphertexts and string search by looking for patterns in the entire text using randomized equality circuits using the Razborov Smolensky method for calculating OR. The pattern is chosen smaller than the slot size and repeated until the slot is filled as this count is maintained. Equality with a wildcard along with randomization also works because of the way the slots are created.

Iliashenko et al. [15] study arithmetic circuits over finite fields representing non-arithmetic functions over integers, thus leading to practically efficient homomorphic implementations of useful algorithms. Along the same lines, Tan et al. [21] compare two field elements based on the lexicographic order of their basis elements for field element encoded items. This encoding is proposed for any cryptosystem that supports RLWE and we use this method in order to improve the comparison circuits in our architecture.

3.2 Protocols for match based search

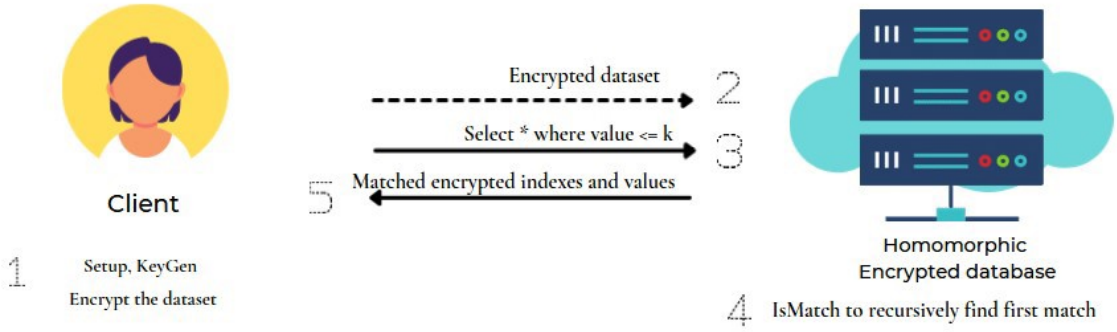

Fig. 1. Search in HE - Standard Model

The standard framework for secure search with data involving only homomorphic encryption as the primitive comes from Akavia et al. [1] where they break Homomorphic search into three steps. The data is stored in an encrypted array which is unsorted. All the schemes that follow their architecture have a single server, one or two rounds of communication, and no setup. The steps may involve some encoding/decoding, compression to reduce workload, etc.

1. Perform *comparison* operations to figure out the matches and store them in the indicator array.
2. Retrieve the matched non-zero indices to get the matched items after getting the indicator array.
3. Use the matched indices to retrieve the elements.

Akavia et al. proposed a way to achieve secure homomorphic search with no pre-processing with a scheme called Spirit in [1] against computationally bounded semi-honest adversaries. Their scheme is agnostic to the FHE as it works on $F(2)$ instead of $F(p)$. Most schemes are mentioned below can be operated on any HE cryptosystem. Search is based on evaluating a polynomial whose roots are the query points. If a ciphertext solves the polynomial, it is included in the result set. They use data summarizing techniques called sketches, where exact first match and retrieval are done using rank and tree matrices, then a step function to find the first match and then retrieves this index, which gets decrypted and sent back. They use Razborov Smolenski-OR instead of logical OR to speed up computation. The security game is to prove IND-Q with equally sized queries.

Overall, they propose a poly-logarithmic search algorithm with extra functionalities like wildcard, range, search in sub-array, sequential retrieval, and boolean logic queries. Their deterministic variant provides good results but is not secure. However, they have a noticeable error in the randomized variant so not feasible. Akavia et al. followed their previous work by increasing the security components in [2]

with same spirit algorithm but instead of the deterministic method, they randomize with binary raffle which is a randomized Monte Carlo algorithm that allows for sequential retrieval. They use a Toeplitz matrix to randomize evenly and they construct hashes for faster matching.

Building on the previous works, Wen et al. build LEAF [22] with a three-step process: *Localize, Extract and Reconstruct*. Localize gets the first matches iteratively and forms an indicator index. They divide the matched index into smaller equal intervals for local search. Then Extract is used to get the interval containing the first non-zero match. Reconstruct the indices back obliviously. LEAF only uses $O(n_c)$ multiplications for search where n_c is the size of the input array. We use LEAF in our architecture in order to perform secure search. Choi et al. [12] proposed some cost-effective additions to this standard protocol by utilizing Bloom filters or power sums to store the data and retrieve obliviously using SealPIR. They introduce new ways to encode data for faster retrieval. Overall, their scheme is faster than LEAF because they do not require multiplications.

4 Our scheme

Our aim is to achieve shared secure search with multiple parties with similar requirements to the previous state-of-the-art ([22], [1]) with no pre-processing and a single one server architecture. We would like to use the properties of optimized multiplications with VFE in the multi-key setting. The first step is to combine MKHE schemes with the VFE encoding. We use MKBFV to illustrate the encoding because it inherently supports packing. Then we run set-up on this scheme and generate a VFE encoded database that is then encrypted.

We consider the MKBFV algorithm along with the BFV encoding in the setup and match computation. Then we can exploit the two methods mentioned below to perform search and retrieval obliviously on the database.

- The state-of-the-art LEAF algorithm to recursively find the matches.
- SealPIR with the decrypted matched indices supplied as input to retrieve the matches obliviously.

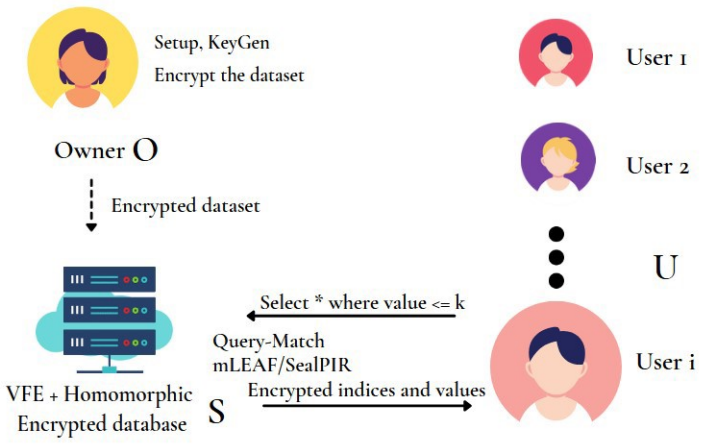

Fig. 2. Secure Search Protocol for Data sharing

4.1 MKHE

We choose the MK-BFV scheme over the other threshold and multikey schemes because it allows packing which enables SIMD and multiple queries and it does not restrict the number of users in set-up, unlike threshold schemes. We need only two parties during computation and we do not need the input of all the other parties. But this can be easily substituted with MKCKKS or MKBGV based

on the application. Because of the cyclotomic polynomial ring used for encoding in these schemes, it is possible to use the VFE encoding to allow for totally ordered and packed ciphertexts.

We do not choose the threshold HE schemes because if we follow the N out of N viable threshold schemes, then we would have to instantiate the database, every time a user joins, to aggregate the public keys. Also, the expansion rate is quadratic in the number of users as they use RGSW ciphertexts, whereas MKFHE has linear expansion.

Chen et al. [9] provide algorithms to convert the extended ciphertexts, which are quadratic to linear ciphertexts using Relinearization algorithms. But scenarios like ours require uneven participation in the computation as one party, the owner, holds more data than the others. MKFHE implies a correspondence between the owner and the N users which can be dynamically adapted as the users enter or leave the computation. In this case, we would like to reduce the size of the extended ciphertexts as it gets quadratic in the number of users after a multiplication. In this case, either the converted key or directly, the ciphertext can be relinearized the ciphertext.

4.2 Set-up

The data owner ($\mathcal{O}$), the trusted party, sends data for search and computation to a *semi-honest server* ($\mathcal{S}$) that can only listen to the queries and results. During Setup, $\mathcal{O}$ encrypts and sends the VFE encoded and encrypted database and evaluation keys $ev_{\mathcal{O}}$ to $\mathcal{S}$. This dataset has to be accessed by non-colluding malicious users $\mathcal{U}$. There is one only round of communication to execute the LEAF protocol between $\mathcal{O}$ and $\mathcal{U}$. We use MKFHE for $\mathcal{O}$ to send public keys and secret keys to the $\mathcal{U}_i$.

The database D has n_c rows and n attributes of the form $\mathbf{D} = \beta_1, \beta_2, ...\beta_{n_c}$, where each $\beta_i = (\beta_{i_1}, \beta_{i_2}, ..., \beta_{i_n})$. $\mathbf{D}$ denoted the encrypted database.

- Generate key pairs individually for the data owner and each user. Then generate the evaluation keys using the secret keys for the relinearization.

$$KeyGen(1^\lambda) \leftarrow (pk_i, sk_i), EvalKeyGen(sk_i, q) \leftarrow ev_i$$

- For the message m and pk_i.

$$Encrypt(pk_i, m) \leftarrow ct_i$$

- Distributed decryption can be illustrated using MKBFV. Given a ciphertext c_i corresponding to user $\mathcal{U}_i$ and their secret s_i, sample an error $e_i \leftarrow \phi$ and return $\mu_i = c_i.s_i + e_i (\text{mod } q)$. Then we merge c_0 with $\mu_i, (1 \leq i \leq k)$ by computing $\mu = c_0 + \sum_{i=1}^{k} \mu_i (\text{mod } q)$ and return $m = \lfloor (t/q)\mu \rceil$

Algorithm 1 Data Owner Set-up

Input: $\mathcal{U}_i$, $\mathbf{D}$

1 In $\mathcal{O}$:
2 Initialize the KeyGen(1^λ) of MKHE to get $sk_{\mathcal{O}}, pk_{\mathcal{O}}, ev_{\mathcal{O}}$
3 For dataset D, encode each β_i using VFE and compute $\mathbf{D} \leftarrow Encrypt(D, pk_{\mathcal{O}})$
4 Send $ev_{\mathcal{O}}$ and $\mathbf{D}$ to the server $\mathcal{S}$
5 For $\mathcal{U}$:
6 Individually every user $\mathcal{U}_i$ runs KeyGen of the MKHE scheme to get sk_i, pk_i, ev_i.

The steps up to the generation of the indicator array a can be adjusted based on any MKFHE scheme we choose, as per Algorithm 1. After Setup, we provide the search protocol for the users. The query $\mathbf{q}_i$ from $\mathcal{U}_i$ consists of many subqueries to be processed simultaneously. We perform the VFE based comparison circuits based on the queries. This implies multiplications and therefore relinearizations. We illustrate this in Alg. 2.

Algorithm 2 Query-match

Input: $\mathbf{q}_i$ from $\mathcal{U}_i$
Output: Indicator array a
1 $\mathcal{U}_i$ sends the query $\mathbf{q}_i$ to $\mathcal{S}$
2 For all $j \in [n_e]$, do $EQ_{\text{VFE}}(\mathbf{q}_i, k_j)$ or $LT_{\text{VFE}}(\mathbf{q}, k_j)$ as per the components of $\mathbf{q}$ and store in $a \in \{0,1\}^n$
3 Perform $\mathbf{q}_i \leftarrow \textbf{Relin}(\mathbf{q}_i, ev_{\mathcal{O}}, b_i)$ to transform $\mathbf{q}$ to same keys of $\mathcal{O}$

4.3 mLEAF

The LEAF algorithm [22] only involves homomorphic encryption and finds a way to do the first match retrieval step in $O(n)$ multiplications unlike Binary Raffle [2] that takes $O(nlogn)$. They also imply that the output of LEAF can be adapted to any retrieval algorithm. They use RS-OR to reduce the number of ORs performed. The final output is the encryption of the bit-representation of the first match index. This is recursively to retrieve all the non-zero indices. We formulate a hybrid between a modified version of LEAF to accommodate [22] the VFE encoding [21] to achieve better performance and security. mLEAF is the modified version of LEAF that works on the VFE encoded data. The input is of the same form, but the multiplications in the middle happen at a different dimension.

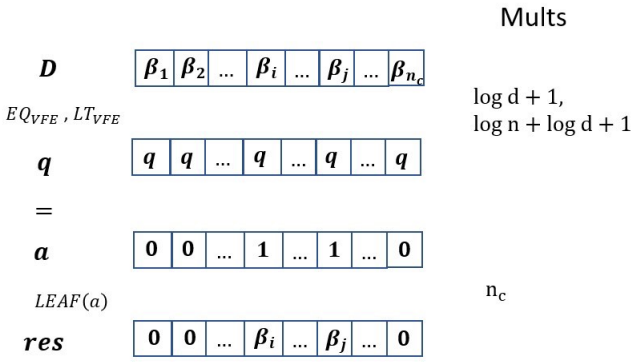

Fig. 3. Search with Query-Match and LEAF

4.4 SealPIR

SealPIR [4] is based on the XPIR algorithm which is the only computationally feasible computational PIR (CPIR). Information-theoretic PIR is computationally expensive and creates multiple copies of the database for security. But CPIR creates queries that are proportional to the size of the database and this provides the security along with the shuffling. In XPIR, the query sent by the client contains one ciphertext (encrypting 0 or 1) for each entry in an n-element database. It works on probabilistic batch codes by separating the database into d-dimensional hypercubes that are made of m codewords and b buckets. The requirement for preprocessing leads to more overhead for the client, along with network costs being a major drawback.

To avoid the search and access pattern attacks possible with a semi-honest server, we use SealPIR to secure the value retrieval process. They use the FV [13] cryptosystem to implement their scheme and this use of HE ensures that the server learns nothing more from the search than the volume. It also allows the retrieval of multiple queries simultaneously. We use SealPIR for the retrieval part because it is highly efficient and secure and is even better than ORAM-based schemes which increase the depth by $logn$. SealPIR does not reveal access or search patterns to the server.

4.5 Secure Search with Finite field extensions

We put the components together in order to create the search protocol. Following Setup, firstly the Query-Match algorithm is executed to provide the indicator array. Then LEAF finds the non-zero indices. It takes care of the entire search algorithm within the server and sends the retrieved indices to SealPIR, which requires an extra round of communication in order to receive the decrypted ciphertexts to be sent back to the server in a particular encoding. Finally, the encrypted results are sent to the user who decrypts them using their keys and the partial shares. We consolidate the process in Alg. 3

4.6 Correctness

All the parties use the same public parameters for KeyGen and therefore can be used to compute between each other and then decrypted separately using partial decryption. The encoding does not change the properties of the original MKHE scheme no matter which (RLWE-based) cryptosystem is used. There is no necessity to have the converted key as it will increase the storage quadratically. Therefore, after a multiplication, one can simply relinearize and have the converted key.

The EQ_{VFE} circuit comprises many EQ_{FE} which works in the following manner with respect to BFV. The ciphertexts are encrypted under different keys but under subtraction, the keys are not compounded. This circuit shall give an encrypted 0 or 1 at the end of the calculation, which is then multiplied $logn + 1$ times. After every multiplication, there is a relinearization to reduce the ciphertext size.

The input from the VFE-MKHE algorithm works with mLEAF. We do not encode the input indicator array for the mLEAF algorithm. This boolean array must therefore be accepted as input without any problems. Therefore, mLEAF still takes $O(n_c)$ multiplications.

The input from the set-up algorithm is acceptable for the SealPIR and does not leak anything because the decrypted indices from the user are sent back from the client to the server. The algorithm masks the indices by encrypting them in a special format to indicate that a non-zero index i is encrypted as the polynomial x^i. This polynomial further gets expanded to perform PIR.

Algorithm 3 Secure Search

Input: $\bar{\mathbf{D}}$, Query vector $\mathbf{q}_i$, $|\mathbf{q}_i| = d$, from $\mathcal{U}_i$
Output: Result set **rs**

1 In $\mathcal{S}$:
2 $a \leftarrow$ Query-Match($\mathbf{q}_i$)
3 Perform $index \leftarrow$ **mLEAF**(a) to get the matched indices and retrieve results as **res**
4 $\mathcal{O}$ performs relinearization and partial decryption on $index$ in $\mathcal{S}$
5 Send **index** and the partial share to $\mathcal{U}_i$
6 $\mathcal{U}_i$ performs partial decryption and merge to get the decrypted indices, then encodes the indices for SealPIR and send it to $\mathcal{S}$.
7 Perform **res** $\leftarrow$ **SealPIR**($index$, **D**) for the encrypted result set.
8 Send **rs** to $\mathcal{U}_i$ along with the secret share of $\mathcal{O}$
9 $\mathcal{U}_i$ decrypts **rs** to get the search result.

5 Security

Leakage in searchable encryption also applies to HE-based search [14] in relation to the search, access, and volume pattern of the data being revealed during the process. The semi-honest server can know if the same query was sent again if it retrieves the data from the same addresses. This is not a desired property. Though HE protects data privacy, typical schemes do not have the properties of obliviousness to hide the metadata like logs and cache. Binary raffle [1] uses the raffle system to maintain obliviousness when retrieving data. COIE [12] uses SealPIR to make the retrieval oblivious but a man in the middle attack can supply the plaintext indices to the second round of the protocol with SealPIR and manage to retrieve some information.

Our scheme addresses these limitations by using the three components which provide security and more functionality with the data at hand. We now provide informal proofs for the components as the

primitives in our architecture do not deviate from their original form. Hence, the security proof for the multi-key scheme, mLEAF, and SealPIR follow the original work. The steps happen sequentially and even though there is a communication overhead because of this, the security is not compromised for the data or the query.

Kim et. al. [17] first proposed a security definition for private database querying using *indistinguishability* of results and queries using the simulation paradigm. Our scheme is secure under the following indistinguishability game. Given a query and two databases, the probability of distinguishing which database the result set game is from is negligible. The same argument can also be given as a database with two queries and finding which query's the result set is, as described below. The challenger can be either the Owner or one of the users. The adversary controls the semi-honest server.

1. The challenger $\mathcal{C}$ runs a KeyGen and sends evk to A so that A can perform homomorphic operations.
2. A chooses either one of the two equivalent scenarios where the sizes of the two result sets are equal.
   - Two databases D^0 and D^1 of the same size, and a query q or
   - A single database D and two queries q_0, q_1 of the same circuit size
3. $\mathcal{C}$ samples $b \leftarrow \{0, 1\}$ and either
   - Runs Setup on D^b and the secure search on q or
   - Runs Setup on D and the secure search on q^b
4. A outputs a bit b' as the guess
5. We say the scheme is secure against the semi-honest adversary if A has negligible advantage in the security parameter.

$$Adv(A) = |Pr[b = b'] - 1/2| \leq negl(\lambda)$$

FHE cryptosystems that support packing like BFV, BGV, and CKKS are IND-CPA secure under the RLWE assumption in the public parameter. MKFHE having the same single-key encryption algorithms implies that the security relies on the hardness of the same RLWE problem. It is proved by showing that the distribution that KeyGen, Setup, and EvalkeyGen are computationally indistinguishable from the uniform distribution over $R_q^d \times R_q^d \times R_q^{d\times 3}$ and the circular security assumption.

The LEAF algorithm takes place only in the server and finds the non-zero indices iteratively. Though it is not ideal in terms of computation, the aim is to reduce the cost and time from the client's side. Therefore, LEAF becomes an ideal candidate to retrieve the non-zero indices. The parameters have to be chosen for better performance with LEAF if the packing has to be included because we use the VFE encoding.

We use SealPIR to obfuscate the addresses of the retrieved elements to prevent leakage and promote oblivious retrieval. SealPIR is effective against semi-honest servers by the computational PIR assumption operating on the probabilistic batch codes and BFV cryptosystem [13]. The input for SealPIR comes from the decrypted indices of the client. Therefore, the security of SealPIR does not have to be composed of the other components.

If we consider other solutions in terms of computation, T out of N threshold FHE schemes can be considered, where all parties compute on ciphertexts encrypted under an aggregated public key that can be decrypted by minimum T parties. In our case, T=2. But this means that the output of the search query can be decrypted by any of the other parties unless the data owner verifies that it was the same ID from which the query came. Also, if the Data owner has a trusted third party that retains all the secret keys, that would cut down the computational cost the most but it would also compromise user privacy. The owner would have N extended keys and N copies of the database. This is not very desirable but performs the tasks in one round of communication.

Overall, our scheme does not reveal the search and access patterns of the queries. The server cannot differentiate between two ciphertexts. The users can be malicious and non-colluding with the server because only the data owner can alter the database. This allows for confidential data to be outsourced and searched in an untrusted cloud by multiple users. This searched data can also be extended to computation because of homomorphic encryption and the relinearization properties.

6 Performance analysis

In terms of performance, multiplication is the most expensive arithmetic homomorphic operation and so, the aim is to reduce the multiplicative depth of the circuits involved in the algorithms. The use

of encrypted arrays as the base data structure implies that we cannot avoid linear search. Sub-linear search requires pre-processing of the data to form some form of an index like inverted index, hash tables, etc. In order to improve security, we require more components like SealPIR that increase the computational cost and time.

Tan et al.. [21] use key-value pairs (a_i, b_i) to illustrate their work. The multi-query with q_r queries is of the form "SELECT $*$ FROM **D** WHERE $a_{i,j} = \beta_{i,j}$ and ..." for $j \in [q_r]$. They provide only a naive retrieval algorithm for their queries without any security guarantees. Our work improves the security of schemes using this special encoding technique by providing obliviousness and access pattern security.

We do not consider the computational cost of *Setup* (Alg. 1) as it may vary when the schemes change. Previous works consider F_2 because the scheme becomes agnostic to the HE cryptosystem used and homomorphic comparisons are considerably faster. But we consider over higher primes in order to enable the encoding and the packing. Relinearization is expensive and requires $O(d)$ multiplications.

After the query is sent, the comparison circuits are executed (Alg. 2). Without the VFE encoding, n multiplications are required per equality. Query-Match requires only $logn$ multiplications per one equality calculation. Other comparisons like Less Than are possible because of the decomposition in the field encoding method. LT takes $O(logn)$ considering that $d < n$. The multiplications here have to be relinearized to enable partial decryption later.

The LEAF algorithm takes n_c multiplication in the flagging step for non-zero elements. This is executed iteratively based on the number of non-zero entries. Therefore, it takes $O(n_c)$ to execute the LEAF step and retrieve the elements. A relinearization may again be required to reduce the size of the ciphertext here. To accommodate a large volume of data, all data items can be 16 or 64-bit integers. SealPIR, even though it has high communication overhead, does not require multiplications. Therefore, the retrieval only involves the communication cost (Alg. 3).

The number of rounds of communication increases with the security of the scheme. There is one round for the query to be sent and to receive back the partial decryption and the *index* array from LEAF. After sending these partial decryptions and merging in the user machine, it takes $O(q_r \sqrt[q_r]{n_c})$ to send back for the EXPAND operation in the server for SealPIR. As we only consider two parties out of the total users for any computation, we reduce the overhead there.

Inserts can be performed by adding the element at the end of the array. Deletes are performed by substitution with 0 and making it a dummy block and then push to the end of the array. Updates can be done by performing a delete and then a fresh insert at the end. Conjunctive and disjunctive queries can be easily introduced into the architecture.

As no open-source implementation for MKBFV or MKCKKS is available, we do not provide here an empirical computational cost analysis. But in Table 1, we provide the theoretical improvements with the use of the state-of-the-art components. We compare the same search algorithms mentioned above, with VFE encoding and without. Operations like Less Than cannot be directly performed on polynomial encodings and therefore, our scheme performs better than normal batching.

Table 1. End-to-end Secure search

		Search	**Batched search**	**Search + VFE**
Setup	Cost	$O(n_c * n * d)$	$O(n_c * n)$	$O(n_c * n)$
Search	Mult EQ	$O(n_c * n)$	$O(n_c * n)$	$O(n_c * logn)$
	Mult LT	$O(n_c * n)$	-	$O(n_c * logn)$

7 Conclusion and Future works

In this paper, we provide a new Multikey homomorphic secure search scheme by combining the most cost-effective multi-key FHE scheme along with the search scheme with the least multiplications. So far, the search in the data sharing scenario has not been explored because of the cost and the possible leakage. We optimize every step of the computation to what the state of the art allows. Our scheme can be applied practically without much difficulty using one of the popular homomorphic encryption libraries with the right parameters. Also, this can be used as a step when AI algorithms are to be

executed on a subset of the encrypted data in the cloud. In this case, instead of SealPIR sending the data back to the client, one can execute neural networks or machine learning algorithms on it. It is also an essential component in clustering related problems to group related data together.

We plan to extend this work by using methods to transform the ciphertext from one key to another. This way one can just change the query to the Owner's keys and switch the query results back to the user's keys in the end. But we have to make sure that the computability of the ciphertexts is not affected during the process. This would also reduce the rounds of communication. The computation and communication cost for this scheme which uses the state of the art already implies that this area requires more focus in order to enable secure outsourcing and sharing of data in the near future.

References

1. Akavia, A., Feldman, D., Shaul, H.: Secure search via multi-ring fully homomorphic encryption. IACR Cryptol. ePrint Arch. p. 245 (2018), http://eprint.iacr.org/2018/245
2. Akavia, A., Gentry, C., Halevi, S., Leibovich, M.: Setup-free secure search on encrypted data: Faster and post-processing free. Proc. Priv. Enhancing Technol. **2019**(3), 87–107 (2019). https://doi.org/10.2478/popets-2019-0038, https://doi.org/10.2478/popets-2019-0038
3. Aloufi, A., Hu, P., Song, Y., Lauter, K.E.: Computing blindfolded on data homomorphically encrypted under multiple keys: An extended survey. CoRR **abs/2007.09270** (2020), https://arxiv.org/abs/2007.09270
4. Angel, S., Chen, H., Laine, K., Setty, S.T.V.: PIR with compressed queries and amortized query processing. In: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA. pp. 962–979. IEEE Computer Society (2018). https://doi.org/10.1109/SP.2018.00062, https://doi.org/10.1109/SP.2018.00062
5. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7237, pp. 483–501. Springer (2012). https://doi.org/10.1007/978-3-642-29011-4_29, https://doi.org/10.1007/978-3-642-29011-4_29
6. Boneh, D., Gentry, C., Halevi, S., Wang, F., Wu, D.J.: Private database queries using somewhat homomorphic encryption. In: Jr., M.J.J., Locasto, M.E., Mohassel, P., Safavi-Naini, R. (eds.) Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7954, pp. 102–118. Springer (2013). https://doi.org/10.1007/978-3-642-38980-1_7, https://doi.org/10.1007/978-3-642-38980-1_7
7. Bonte, C., Iliashenko, I.: Homomorphic string search with constant multiplicative depth. In: Zhang, Y., Sion, R. (eds.) CCSW'20, Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop, Virtual Event, USA, November 9, 2020. pp. 105–117. ACM (2020). https://doi.org/10.1145/3411495.3421361, https://doi.org/10.1145/3411495.3421361
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. Electron. Colloquium Comput. Complex. p. 111 (2011), https://eccc.weizmann.ac.il/report/2011/111
9. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 395–412. ACM (2019). https://doi.org/10.1145/3319535.3363207, https://doi.org/10.1145/3319535.3363207
10. Chen, L., Zhang, Z., Wang, X.: Batched multi-hop multi-key FHE from ring-lwe with compact ciphertext extension. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10678, pp. 597–627. Springer (2017). https://doi.org/10.1007/978-3-319-70503-3_20, https://doi.org/10.1007/978-3-319-70503-3_20
11. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) 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. Lecture Notes in Computer Science, vol. 10624, pp. 409–437. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_15, https://doi.org/10.1007/978-3-319-70694-8_15
12. Choi, S.G., Dachman-Soled, D., Gordon, S.D., Liu, L., Yerukhimovich, A.: Compressed oblivious encoding for homomorphically encrypted search. In: Kim, Y., Kim, J., Vigna, G., Shi, E. (eds.) CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea,

November 15 - 19, 2021. pp. 2277–2291. ACM (2021). https://doi.org/10.1145/3460120.3484792, https://doi.org/10.1145/3460120.3484792
13. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptol. ePrint Arch. p. 144 (2012), http://eprint.iacr.org/2012/144
14. Ganesh, B., Palmieri, P.: A survey of advanced encryption for database security: Primitives, schemes, and attacks. In: Nicolescu, G., Tria, A., Fernandez, J.M., Marion, J., García-Alfaro, J. (eds.) Foundations and Practice of Security - 13th International Symposium, FPS 2020, Montreal, QC, Canada, December 1-3, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12637, pp. 100–120. Springer (2020). https://doi.org/10.1007/978-3-030-70881-8_7, https://doi.org/10.1007/978-3-030-70881-8_7
15. Iliashenko, I., Zucca, V.: Faster homomorphic comparison operations for BGV and BFV. Proc. Priv. Enhancing Technol. **2021**(3), 246–264 (2021). https://doi.org/10.2478/popets-2021-0046, https://doi.org/10.2478/popets-2021-0046
16. Kim, H., Kim, H., Chang, J.: A secure knn query processing algorithm using homomorphic encryption on outsourced database. Data Knowl. Eng. **123** (2019). https://doi.org/10.1016/j.datak.2017.07.005, https://doi.org/10.1016/j.datak.2017.07.005
17. Kim, M., Lee, H.T., Ling, S., Ren, S.Q., Tan, B.H.M., Wang, H.: Better security for queries on encrypted databases. IACR Cryptol. ePrint Arch. p. 470 (2016), http://eprint.iacr.org/2016/470
18. Kim, M., Lee, H.T., Ling, S., Tan, B.H.M., Wang, H.: Private compound wildcard queries using fully homomorphic encryption. IEEE Trans. Dependable Secur. Comput. **16**(5), 743–756 (2019). https://doi.org/10.1109/TDSC.2017.2763593, https://doi.org/10.1109/TDSC.2017.2763593
19. Kim, M., Lee, H.T., Ling, S., Wang, H.: On the efficiency of fhe-based private queries. IEEE Trans. Dependable Secur. Comput. **15**(2), 357–363 (2018). https://doi.org/10.1109/TDSC.2016.2568182, https://doi.org/10.1109/TDSC.2016.2568182
20. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptogr. **71**(1), 57–81 (2014). https://doi.org/10.1007/s10623-012-9720-4, https://doi.org/10.1007/s10623-012-9720-4
21. Tan, B.H.M., Lee, H.T., Wang, H., Ren, S.Q., Aung, K.M.M.: Efficient private comparison queries over encrypted databases using fully homomorphic encryption with finite fields. IEEE Trans. Dependable Secur. Comput. **18**(6), 2861–2874 (2021). https://doi.org/10.1109/TDSC.2020.2967740, https://doi.org/10.1109/TDSC.2020.2967740
22. Wen, R., Yu, Y., Xie, X., Zhang, Y.: LEAF: A faster secure search algorithm via localization, extraction, and reconstruction. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020. pp. 1219–1232. ACM (2020). https://doi.org/10.1145/3372297.3417237, https://doi.org/10.1145/3372297.3417237
23. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Secure pattern matching using somewhat homomorphic encryption. In: Juels, A., Parno, B. (eds.) CCSW'13, Proceedings of the 2013 ACM Cloud Computing Security Workshop, Co-located with CCS 2013, Berlin, Germany, November 4, 2013. pp. 65–76. ACM (2013). https://doi.org/10.1145/2517488.2517497, https://doi.org/10.1145/2517488.2517497