On the Efficiency of FHE-based Private Queries*

Myungsun Kim [1], Hyung Tae Lee [2], San Ling [2], and Huaxiong Wang [2]

[1] Department of Information Security
The University of Suwon, South Korea
msunkim@suwon.ac.kr
[2] Division of Mathematical Sciences
School of Physical & Mathematical Sciences
Nanyang Technological University, Singapore
{hyungtaelee, lingsan, hxwang}@ntu.edu.sg

Abstract. Private query processing is a very attractive problem in the fields of both cryptography and databases. In this work, we restrict our attention to the efficiency aspect of the problem, particularly for basic queries with conditions on various combinations of *equality*. Without loss of generality, these conditions can be regarded as a Boolean function, and this Boolean function can then be evaluated at ciphertexts produced by a fully homomorphic encryption (FHE) scheme *without decryption*. From the efficiency perspective, the remaining concern is to efficiently test the equality function without severely downgrading the performance of FHE-based querying solutions.

To this end, we first analyze the multiplicative depth required for an equality test algorithm with respect to the plaintext space inhabited by general FHE schemes. The primary reason for this approach is that given an equality test algorithm, its efficiency is measured in terms of the multiplicative depth required to construct its arithmetic circuit expression. Indeed, the implemented equality test algorithm dominates the entire performance of FHE-based query solutions, apart from the performance of the underlying FHE scheme. Then, we measure the multiplicative depth considering an FHE scheme that takes an extension field as its plaintext space and that supports the depth-free evaluation of Frobenius maps. According to our analysis, when the plaintext space of an FHE scheme is a field of characteristic 2, the equality test algorithm for ℓ-bit messages requires the lowest multiplicative depth $\lceil \log \ell \rceil$. Furthermore, we design a set of private query protocols for conjunctive, disjunctive, and threshold queries based on the equality test algorithm. Similarly, applying the equality test algorithm over $\mathbb{F}_{2^\ell}$, our querying protocols require the minimum depths. More specifically, a multiplicative depth of $\lceil \log \ell \rceil + \lceil \log(1+\rho) \rceil$ is required for conjunctive and disjunctive queries, and a depth of $\lceil \log \ell \rceil + 2\lceil \log(1+\rho) \rceil$ is required for threshold conjunctive queries, when their query conditions have ρ attributes to be compared. Finally, we provide a communication-efficient version of our solutions, though with additional computational costs, when an upper bound δ $(0 \leq \delta \leq 1)$ on the selectivity of a database is given. Consequently, we reduce the communication cost from n to approximately $\lfloor \delta n \rfloor$ ciphertexts with $\lceil \log n \rceil$ additional depth when the database consists of n tuples.

Keywords: Encrypted databases, Homomorphic encryption, Private queries

1 Introduction

Group homomorphic encryption (GHE) schemes, such as the Paillier and ElGamal cryptosystems, allow group operations to be performed on encrypted data without decryption. Many applications, such as private SQL query processing, utilize GHE to prevent breaches of personal data by strong attackers, such as corrupt employees. However, because GHE does not support ring operations on ciphertext domains, there are intrinsic restrictions on the range of functions that are directly computable over ciphertexts.

The past few years have witnessed significant progress in ring-based cryptosystems, relieving these limitations, since Gentry's proposal [1] of fully homomorphic encryption (FHE). In principle, FHE schemes can evaluate arbitrary functions over encrypted data. One of the most

* Published in *IEEE Transactions on Dependable and Secure Computing* (DOI: 10.1109/TDSC.2016.2568182).

promising applications of such schemes is the ability to realize a *search* functionality without losing data confidentiality. The first concrete implementation of this functionality using a specific FHE scheme was presented by Boneh et al. [2]. Their protocol allows users to submit an SQL query and to obtain only a set of matching results. Kissner-Song's set intersection protocol [3] can be used as a sub-protocol through combination with Brakerski's somewhat homomorphic cryptosystem [4]. However, their protocol allows only the retrieval of queries with conjunctive conditions.

Quite recently, Cheon et al. [5] attempted a different approach to obtain a more *general* solution to the FHE-based search problem. The basic idea is to find a predicate to efficiently represent a search condition (e.g., equality test and greater-than comparison) and then to evaluate that predicate at each FHE ciphertext. It is obvious that the evaluation result is also an encryption of one or zero. For instance, consider a predicate `EQTest` for testing the equality of two ciphertexts, c_1 and c_2. If the plaintext of c_1 is equal to that of c_2, then `EQTest` outputs an encryption of 1; otherwise, it outputs an encryption of 0.

Then, applying a few additional computations to the equality test produces only the search result of interest. Consider a list of ciphertexts $\bar{\mathsf{R}} = \{c_1, \ldots, c_n\}$, where $c_i = \mathsf{E}(a_i)$ for a plaintext message a_i and an FHE encryption algorithm E. For a fixed value v, anyone can then compute

$$\langle c_i \cdot \texttt{EQTest}(c_i, \mathsf{E}(v)) \rangle_{i=1}^{n} .$$

Under the assumption that a relation $\mathsf{R}(A)$ has a state $r(\mathsf{R}) = \{a_1, \ldots, a_n\}$ and that $\bar{\mathsf{R}}$ in the clear is equal to $r(\mathsf{R})$ as a set, the expression is semantically equivalent to the following pseudo-SQL expression:

$$\texttt{select} * \texttt{from}\ \mathsf{R}\ \texttt{where}\ A = v.$$

Arguably, this FHE-based technique allows any SQL statement to be privately processed.

In their solution, the efficiency of the implemented predicates dominates the entire performance of the query processing system. Of course, one may regard the efficiency of the underlying FHE scheme as more important. This topic is of interest in its own right but is beyond our scope. Thus, the next step naturally appears to be to explore efficient predicates for SQL query processing, their performance, and their further optimization within an FHE context. These issues were broadly addressed in Cheon et al.'s work. Still, we have identified several points that the authors missed. The first is that Cheon et al.'s narrow choice for the plaintext domain does not harmonize well with their goal of generality. Specifically, their work focused on the plaintext domain $\mathbb{F}_2$. Of course, Cheon et al. also considered $\mathbb{F}_{2^\ell}$, but the aim of such consideration was to reduce the required multiplicative depth for an addition algorithm by applying the lazy carry operation technique, and merely considered a small integer ℓ. The reason for focusing on the plaintext space is that the efficiency of a predicate is measured in terms of its multiplicative depth in Boolean circuit form, which is, in turn, strongly related to the plaintext space.

The second is that the communication cost for search operations is exactly n times the ciphertext size, where n is the dataset size. Interestingly, although Boneh et al.'s protocol supports only conjunctive queries, it returns a set of exactly matching results. Namely, for a database of selectivity δ, the communication cost decreases to δn times the ciphertext size. We will thus attempt to reduce the communication cost of the general solution by encoding the resulting output as a polynomial.

Throughout the remainder of this work, we focus on the predicate for an `equality` *test* because it can cover most basic SQL retrieval queries. However, we believe that our analysis can also be extended to address the efficiency of different predicates, such as `greater-than` comparisons and `min/max` computations on ciphertexts.

1.1 Our Results

In this work, our contributions are three-fold: (1) we analyze the multiplicative depth of an equality test algorithm associated with the specific plaintext space of an FHE scheme, (2) we design a series of private query techniques based on an equality test algorithm with the minimum depth, and (3) finally, we devise an efficient variant of our solutions from the perspective of communication cost on the server side. More concretely, we can rephrase these contributions as follows.

- We analyze the multiplicative depth for running an equality test algorithm[3] based on the plaintext message space inhabited by the FHE schemes. We consider plaintext spaces of $\mathbb{F}_2$, $\mathbb{F}_{2^\ell}$, $\mathbb{F}_p$, and $\mathbb{F}_{p^\ell}$, where p is prime and ℓ is a positive integer.
  Specifically, we first demonstrate the minimum multiplicative depth for an equality test with respect to the plaintext spaces. We then show how to reduce the required depth, provided that we can adopt a means of Frobenius map evaluation whose computation does not consume multiplicative depth. According to our analysis, when an FHE scheme takes a plaintext space as a field of characteristic 2, the equality test algorithm requires the lowest multiplicative depth.
- Based on an equality test algorithm, we design a set of private query processing protocols for conjunctive, disjunctive, and threshold conjunctive queries. In addition, we investigate the total multiplicative depth required for the protocols to function correctly.
- The key concept for achieving privacy in FHE-based query applications is to cause all ciphertexts whose corresponding plaintext do not satisfy a given condition to carry encryption of 0 by multiplying by a zero encryption. As a trade-off, all resulting outputs must be sent because it is impossible to know whether any particular output is an encryption of 0. Given an upper bound δ on the selectivity of a database with n elements, we show that using polynomial representations makes it possible to transmit only approximately δn ciphertexts rather than n ciphertexts at the expense of an additional multiplicative depth of $\lceil \log n \rceil$. Although an overhead of ciphertext expansion arises when using larger depth FHE schemes, our improvement appears to perform sufficiently well in practice. According to our estimation in Section 5, our proposed approach reduces the communication cost by about 80 % when $n = 2^{20}$ and $\delta = 0.05$.

1.2 Related Work

To our knowledge, few FHE-based private query solutions have been reported in the literature. The primary reason for this lack is that the efficiency of known FHE schemes has been the primary concern and these schemes are assumed to be far from practical usage. However, their efficiency has been significantly improved by various researchers (e.g., [6, 7]).

Before the end of this section, it is worthwhile to briefly consider the available non-FHE-based approaches for private query processing. We simply provide a short summary of the most recent results. Of particular interest among these results are CryptDB [8] and its extension, called Monomi [9]. These approaches involve the repeated application of cryptographic tools (e.g., AES, PRF, OPE,[4] and GHE) in different manners depending on the properties of the data. Hence, it is difficult to estimate the security of their schemes in the cryptographic sense. The cryptography literature also contains many intensive studies in this area. Searchable encryption [10] is one leading example. One can also find examples based on private information

[3] The equality test must be expressed in the form of a Boolean or arithmetic circuit prior to being employed in real applications. We call a circuit representation of the equality predicate an equality test algorithm. However, we will sometimes use the two terms interchangeably when no confusion can arise.

[4] PRF and OPE are abbreviations for pseudorandom function and order-preserving encryption, respectively.

retrieval, predicate encryption, or oblivious RAM. However, none of them can achieve all desired goals with only a single cryptosystem, as can be done in the FHE-based approach.

The structure. The remainder of the paper is organized as follows. In Section 2, we present some background on the problem. Section 3 provides the detailed analysis of the multiplicative depth of an equality test on each underlying plaintext space. Using this result, in Section 4, we construct a set of private query protocols for various equality conditions. Section 5 shows how to improve the communication cost of our work, followed by our conclusion in Section 6.

2 Basics and Definitions

Notation. Throughout the paper, $\lceil a \rceil$ (resp. $\lfloor a \rfloor$) denotes the smallest (resp. largest) integer that is larger (resp. smaller) than or equal to a real number a. For an integer a, $[a]$ denotes the set of positive integers from 1 to a. The cardinality of a set S is denoted by $|S|$. Let $\bar{a}$ denote an encryption of a plaintext message a, omitting its randomness. The set of positive integers is denoted by $\mathbb{Z}_+$.

2.1 Fully Homomorphic Encryption

An FHE scheme, denoted by $\mathsf{FHE} = (\mathsf{Kg}, \mathsf{E}, \mathsf{D}, \mathsf{Ev})$, is a quadruple of probabilistic polynomial-time algorithms as follows.

– *Key generation.* This algorithm takes the security parameter λ and outputs a public encryption key pk, a public evaluation key ek, and a secret decryption key sk. We write the algorithm as $(pk, ek, sk) \leftarrow \mathsf{Kg}(1^\lambda)$ and assume that the public key specifies both the plaintext space $\mathcal{P}$ and the ciphertext space $\mathcal{C}$.
– *Encryption.* The algorithm $\bar{a} \leftarrow \mathsf{E}_{pk}(a)$ takes the public key pk and a message $a \in \mathcal{P}$ and outputs a ciphertext $\bar{a} \in \mathcal{C}$.
– *Decryption.* The algorithm $a^* \leftarrow \mathsf{D}_{sk}(\bar{a})$ takes the secret key sk and a ciphertext $\bar{a}$ and outputs a message $a^* \in \mathcal{P}$.
– *Homomorphic evaluation.* The algorithm Ev takes the evaluation key ek, a function $\varphi : \mathcal{P}^n \to \mathcal{P}$, and a set of n ciphertexts $\bar{a}_1, \ldots, \bar{a}_n$ and outputs a ciphertext $\bar{a}_\varphi$.

FHE algorithm selection. FHE schemes (e.g., [11, 12]) that follow Gentry's design philosophy have fairly poor performance. A later series of results were proposed to address this concern. In particular, Brakerski and Vaikuntanathan introduced the notion of leveled FHE, which allows the evaluation of arbitrary arithmetic circuits of polynomial depth [13]. Following this proposal, Brakerski, Gentry, and Vaikuntanathan (BGV) in [14] further presented a leveled FHE scheme with significantly improved performance.

For certain performance-critical systems, we recommend somewhat homomorphic encryption (SHE) if possible, but if circumstances do not permit this approach, an efficient leveled FHE scheme should be considered while minimizing the multiplicative depth of the target algorithms. Because an equality test algorithm can take as input a large plaintext message, the minimum requirement for the implementation of a private query system is to use an efficient leveled FHE scheme with depth-free automorphisms, and further optimization is recommended. (See the HElib library [15]).

In principle our proposed protocols work by repeatedly invoking an equality test algorithm as a sub-routine. Different from our choice, one may implement an algorithm for equality test based on SHE. All currently existing SHE and (leveled) FHE schemes are constructed by following Gentry's strategy [1] that is to insert small noise into a ciphertext and so noise in a result of the evaluation algorithm grows up. While the noise growth in SHE schemes is exponential in the number of multiplications, that in leveled FHE schemes is polynomial. Thus a leveled FHE is a better choice for efficient implementations of our protocols.

2.2 Polynomial Representation of Sets

Given a set, using its equivalent polynomial rather than the set itself is a standard technique in cryptography, particularly in the privacy-preserving literature.

Let R be a ring. For a set $S = \{s_1, s_2, \ldots, s_n\}$ whose elements are in R, a polynomial representation of S is defined by

$$f_S(x) = \prod_{s_i \in S}(x - s_i) \in R[x].$$

Freedman et al. [16] first introduced this representation for the construction of private set intersection protocols. Later, various related methods [3, 17] were proposed to ensure privacy while efficiently addressing set-related problems.

3 Probing the Equality Test

An equality test algorithm (or circuit) is the primitive considered in this work and in other FHE-based private query techniques. Its efficiency is directly related to the efficiency of its higher applications. In this section, we evaluate the multiplicative depth to construct an efficient equality test algorithm for application to encrypted data with various underlying plaintext spaces.

Throughout this section, we use $\mathcal{M}$ to denote a native message space. We intentionally distinguish this space from the plaintext space $\mathcal{P}$ associated with the public key of an FHE scheme. In essence, the algebraic framework of FHE requires a ring $\mathcal{P}$. Here, the key lies in the definition of the native message space $\mathcal{M}$. The general convention is to implicitly assume a map that associates an element in $\mathcal{M}$ with a ring element in $\mathcal{P}$. A typical example of $\mathcal{M}$ is $\{0,1\}^*$, and $\mathbb{F}_2$ is a typical example of the plaintext spaces used in most FHE schemes. Because this work is focused on the plaintext space in relation to the multiplicative depth, it appears preferable to address the native message space separately.

In addition, we prefer the term "algorithm" to the term "circuit" because the latter may cause confusion for cryptographic hardware engineers. If there is no potential for confusion, the reader may interchange the two terms.

3.1 Background

We first define an equality test algorithm for two encrypted data, denoted by $\texttt{EQTest}(\cdot,\cdot)$, as follows: For messages $a_1, a_2 \in \mathcal{M}$,

$$\texttt{EQTest}(\bar{a}_1, \bar{a}_2) := \begin{cases} \bar{1} & \text{if } a_1 = a_2 \\ \bar{0} & \text{otherwise.} \end{cases}$$

This algorithm outputs an encryption of 1 if the two plaintexts of the input ciphertexts are the same and an encryption of 0 otherwise.

3.2 Analysis of the Multiplicative Depth

We continue by constructing an `EQTest` algorithm using arithmetic circuits and present a step-by-step evaluation of the multiplicative depth for this algorithm.

The Case over $\mathbb{F}_2$ For the case in which the plaintext space of an FHE scheme is $\mathbb{F}_2$ and the native message space $\mathcal{M}$ is $\{0,1\}^\ell$ for a fixed positive integer ℓ, several works have already evaluated the multiplicative depth required to perform the `EQTest` algorithm (e.g., [5, 18, 19]).

For a message $a_i = (a_{i\ell}, \ldots, a_{i1}) \in \{0,1\}^\ell$, let us define an encryption of a_i in a bit-by-bit manner, i.e., $\bar{a}_i = (\bar{a}_{i\ell}, \ldots, \bar{a}_{i1})$. According to previous work, for given ciphertexts $\bar{a}_1$ and $\bar{a}_2$, the `EQTest` algorithm can be realized by computing

$$\texttt{EQTest}(\bar{a}_1, \bar{a}_2) = \textstyle\bigwedge_{j=1}^{\ell} \left(\bar{1} + \bar{a}_{1j} + \bar{a}_{2j}\right)$$

because $\bigwedge_{j=1}^{\ell}(1 \oplus a_{1j} \oplus a_{2j}) = 1$ if and only if $a_{1j} = a_{2j}$ for all $j \in [\ell]$. Here, '+' represents the addition of ciphertexts. Because ℓ multiplications between ciphertexts are required to compute the `EQTest` algorithm as above, it consumes a multiplicative depth of $\lceil \log \ell \rceil$.

Supporting Lemmas on Finite Extension Fields In this subsection, we present several theories to prepare for discussing plaintext spaces other than $\mathbb{F}_2$.

Henceforth, let p be a prime, and let $\ell \in \mathbb{Z}_+$. We begin by introducing a new concept regarding a relationship between a polynomial and a function defined over an extension field $\mathbb{F}_{p^\ell}$.

Definition 1 *Let p be a prime, and let $\ell, n \in \mathbb{Z}_+$. For a function $f : (\mathbb{F}_{p^\ell})^n \to \mathbb{F}_{p^\ell}$, we say that a polynomial g is a polynomial expression of f if $g(a_1, \ldots, a_n) = f(a_1, \ldots, a_n)$ for all $(a_1, \ldots, a_n) \in (\mathbb{F}_{p^\ell})^n$.*

The following lemma guarantees the existence and uniqueness of a polynomial expression g of a function $f : (\mathbb{F}_{p^\ell})^n \to \mathbb{F}_{p^\ell}$, under the restriction that the degree of g with respect to each variable is at most $p^\ell - 1$. This lemma is a generalization of Proposition 1 in [20], which addresses the case of a finite prime field, i.e., $\ell = 1$ in our case, and thus, the proof of this lemma is a natural extension of that of Proposition 1 in [20].

Lemma 1 *For any function $f : (\mathbb{F}_{p^\ell})^n \to \mathbb{F}_{p^\ell}$ with p, ℓ, and n as above, there exists a unique polynomial expression g of f whose degree is at most $p^\ell - 1$ with respect to each variable.*

Proof. To prove the existence of g, we define a polynomial $g_a(x)$ as $g_a(x) := \prod_{i=1}^{n} \left(1 - (x_i - a_i)^{p^\ell - 1}\right)$ for any $a = (a_1, \ldots, a_n) \in (\mathbb{F}_{p^\ell})^n$ and variables $x = (x_1, \ldots, x_n)$. Then, we have

$$g_a(b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

by Fermat's Little Theorem on extension fields. We then define $g(x)$ as $g(x) := \displaystyle\sum_{a \in (\mathbb{F}_{p^\ell})^n} g_a(x) f(a)$.
We see that $g(a) = f(a)$ for all $a \in (\mathbb{F}_{p^\ell})^n$ and that the degree of g with respect to each x_i is at most $p^\ell - 1$. Therefore, g is a polynomial expression of f.

To show the uniqueness of g, it is sufficient to consider the case in which $f = 0$. The proof is by mathematical induction on n, the number of variables. We note that the zero polynomial is a trivial polynomial expression of a zero function f.

(i) The case $n = 1$ is that for all $a \in \mathbb{F}_{p^\ell}$, because $g(a) = f(a) = 0$, the polynomial remainder theorem states that $(x - a)$ divides $g(x)$. Hence, $g(x)$ is the zero polynomial, or the degree of $g(x)$ should be at least p^ℓ. Therefore, $g(x)$ is the zero polynomial and the uniqueness holds for $n = 1$.

(ii) Suppose that the uniqueness holds in the case that the number of variables is at most $n-1$. Assume that for the zero function f defined on $(\mathbb{F}_{p^\ell})^n$, there exists a non-zero polynomial $\hat{g}(x_1, \ldots, x_n)$ such that $\hat{g}(a) = f(a) = 0$ for all $a \in (\mathbb{F}_{p^\ell})^n$ and its degree is at most $p^\ell - 1$ with respect to each variable. Then, w.l.o.g, for some integer $0 \le k \le p^\ell - 1$, there exists a non-zero polynomial $\hat{g}_k \in \mathbb{F}_{p^\ell}[x_1, \ldots, x_{n-1}]$ such that $\hat{g} = \sum_{j=0}^{p^\ell-1} \hat{g}_j(x_1, \ldots, x_{n-1}) \cdot {x_n}^j$ with $\hat{g}_j \in \mathbb{F}_{p^\ell}[x_1, \ldots, x_{n-1}]$. However, by considering $\hat{g}$ as a polynomial of one variable x_n, $\hat{g}$ should be the zero polynomial by the same reason in (i) and hence $\hat{g}_k(x_1, \cdots, x_{n-1})$ should be equal to the zero function as a function. Hence, there exists a non-zero polynomial expression $\hat{g}_k$ of the zero function of $n-1$ variables and this is a contradiction with the statement that the uniqueness holds in the case that the number of variables is at most $n-1$.

From (i) and (ii), the uniqueness is therefore proven by mathematical induction on n. $\square$

We call the unique polynomial expression of a function f in Lemma 1 the *minimal polynomial expression* of f. The following lemma shows that such a minimal polynomial expression has the minimum total degree among all polynomial expressions of f. As implied above, Lemma 2 is a generalization of Proposition 2 in [20]. We state the lemma below and provide a formal proof.

Lemma 2 *For a function $f : (\mathbb{F}_{p^\ell})^n \to \mathbb{F}_{p^\ell}$ with p, ℓ, and n as above, the minimal polynomial expression of f has the minimum total degree among all polynomial expressions of f.*

Proof. For a polynomial g, let $\deg(g)$ denote the total degree of g, and let $\deg_{x_i}(g)$ denote the degree of g with respect to a variable x_i. Let g be the minimal polynomial expression of a function f. Assume that a polynomial expression $\hat{g}$ of f has the minimum total degree and that $\hat{g} \neq g$ as polynomials. Because of the uniqueness of the minimal polynomial expression, $\deg_{x_i}(g) \ge p^\ell$ for some $i \in [n]$. Additionally, because ${x_i}^{p^\ell} = x_i$ for each i by Fermat's Little Theorem, we can obtain a polynomial h such that $h = \hat{g}$ as functions on $\left(\mathbb{F}_{p^\ell}\right)^n$ and $\deg_{x_i}(h) < p^\ell$ by repeatedly replacing ${x_i}^{p^\ell}$ with x_i for all $i \in [n]$. Then, $h = g$ because of the uniqueness property, and $\deg(g) = \deg(h) < \deg(\hat{g})$. Therefore, the minimal polynomial expression g has the minimum total degree among all polynomial expressions of f. This completes the proof. $\square$

The multiplicative depth required to evaluate a polynomial at FHE ciphertexts is determined by the total degree of the polynomial. Lemma 2 states that in general, we can evaluate a function with the minimum multiplicative depth by evaluating it in the form of the minimal polynomial expression.

The Case over $\mathbb{F}_{p^\ell}$ Keeping the lemmas in mind, we estimate the multiplicative depth for the `EQTest` algorithm when the plaintext space $\mathcal{P}$ and the native message space $\mathcal{M}$ are both $\mathbb{F}_{p^\ell}$.

We first obtain the minimal polynomial expression of the `EQTest` function as follows:

$$g(x_1, x_2) = 1 - (x_1 - x_2)^{p^\ell - 1} = \begin{cases} 1 & \text{if } x_1 = x_2 \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the `EQTest` algorithm for $\mathcal{M} = \mathbb{F}_{p^\ell}$ can be realized by computing

$$\texttt{EQTest}(\bar{a}_1, \bar{a}_2) = g(\bar{a}_1, \bar{a}_2) = \bar{1} - (\bar{a}_1 - \bar{a}_2)^{p^\ell - 1}$$

for $a_1, a_2 \in \mathbb{F}_{p^\ell}$. We see that a multiplicative depth of $\lceil \log(p^\ell - 1) \rceil$ is required to perform the `EQTest` algorithm in a naïve manner; this quantity is approximately equal to $\lceil \ell \log p \rceil$.

Improvements We can further reduce the multiplicative depth to $\lceil \log(p-1) \rceil + \lceil \log \ell \rceil$ by switching to an FHE scheme that consumes (almost) no depth to evaluate Frobenius maps. For instance, the BGV FHE cryptosystem [14] and YASHE [21] consume no depth to evaluate Frobenius maps; it consists of an exponentiation with exponent p when the BGV encryption scheme and YASHE scheme take $\mathbb{F}_{p^\ell}$ as their plaintext spaces. Using this property, we can improve the multiplicative depth when performing exponentiation over FHE ciphertexts.

Assume that we are using an FHE scheme with the same condition as above; additionally, assume that the FHE scheme may apply Frobenius maps while consuming no depth.

To compute $\bar{a}^e$, we first find the unique p-ary representation of e as $e = \sum_{i=0}^{n} e_i p^i$, where $e_i \in \mathbb{Z}_p$. Then, we evaluate $\bar{a}^e$ as follows:

$$\bar{a}^e = \bar{a}^{\sum_{i=0}^{n} e_i p^i} = \prod_{i=0}^{n} \left(\bar{a}^{p^i}\right)^{e_i}.$$

Because the computation of $\bar{a}^{p^i}$ is depth-free, this requires a multiplicative depth of at most $\log\left(\sum_{i=1}^{n} e_i\right)$ for first computing $\bar{a}^{p^i}$ and then multiplying $\bar{a}^{p^i}$ e_i times for all $0 \leq i \leq n$.

Furthermore, in the case of the `EQTest` algorithm, because $p^\ell - 1$ can be divided by two factors, $(p-1)$ and $(p^{\ell-1} + \cdots + p + 1)$, we can compute $\texttt{EQTest}(\bar{a}_1, \bar{a}_2)$ with a multiplicative depth of $(\lceil \log(p-1) \rceil + \lceil \log \ell \rceil)$ by first computing $\bar{a}^{p-1}$ and then computing an exponentiation with the exponent $(p^{\ell-1} + \cdots + p + 1)$ or vice versa.

Table 1. Comparison of the multiplicative depths required for $\texttt{EQTest}(\cdot,\cdot)$

$\mathcal{M}$	$\mathcal{P}$	Depth-free Frobenius	Multiplicative depth
$\{0,1\}^\ell$	$\mathbb{F}_2$	–	$\lceil \log \ell \rceil$
$\mathbb{F}_p$	$\mathbb{F}_p$	–	$\lceil \log(p-1) \rceil$
$\mathbb{F}_{2^\ell}$	$\mathbb{F}_{2^\ell}$	×	ℓ
		O	$\lceil \log \ell \rceil$
$\mathbb{F}_{p^\ell}$	$\mathbb{F}_{p^\ell}$	×	$\lceil \ell \log p \rceil$
		O	$\lceil \log(p-1) \rceil + \lceil \log \ell \rceil$

$\mathcal{M}$: the native message space, $\mathcal{P}$: the plaintext space

In Table 1, we summarize the multiplicative depths required to perform the `EQTest` algorithm with respect to the native message space and the plaintext space of an FHE scheme as well as the availability of depth-free Frobenius map evaluation. Our analysis shows that when the plaintext space is a field of characteristic 2, the depth required for the `EQTest` algorithm is $\lceil \log \ell \rceil$, and this is the optimal choice among all (existing) plaintext spaces for FHE schemes. For example, when the native message space is $\mathbb{F}_{2^{32}}$ or $\{0,1\}^{32}$, the `EQTest` algorithm consumes a multiplicative depth of 5 when using FHE schemes whose plaintext space is either $\mathbb{F}_2$ or $\mathbb{F}_{2^{32}}$.

4 Return to Private Query Processing

This section presents a series of private querying protocols to support conjunctive, disjunctive, and threshold conjunctive queries and analyzes their required multiplicative depths.

To exploit the best result from Section 3, in Section 4 and Section 5, we assume that a database was encrypted using an FHE scheme with a plaintext space of $\mathbb{F}_{2^\ell}$ for $\ell \in \mathbb{Z}_+$, that supports the depth-free evaluation of Frobenius maps. Henceforth, let $p(x)$ be an irreducible polynomial where $\mathbb{F}_{2^\ell}$ is isomorphic to $\mathbb{F}_2[x]/(p(x))$ and $t \in \mathbb{F}_{2^\ell}$ be a root of $p(x)$.

Remark 1 *We note that one may select an FHE scheme whose plaintext space is $\mathbb{F}_2$, because the required multiplicative depth for an equality test algorithm for ℓ-bit messages is also $\lceil \log \ell \rceil$ in this case. For our conjunctive and disjunctive query protocols, we can easily adapt such FHE schemes and they consume the same multiplicative depth as the case where we use an FHE scheme with the plaintext space $\mathbb{F}_{2^\ell}$. However, for our threshold querying protocol, the server has to evaluate a polynomial (denoted by g in our protocol in Section 4.2) defined over a field, where the number of elements in a field is larger than the number of attributes in the threshold condition, ρ. Since at least $\lceil \log \rho \rceil$ depth is required for addition and multiplication over such a field, our threshold protocol with an FHE scheme whose plaintext space is $\mathbb{F}_2$, spends more multiplicative depths than that with an FHE scheme whose plaintext space is $\mathbb{F}_{2^\ell}$.*

4.1 Background

Additional notation. Throughout Section 4 and Section 5, let $D = \bar{\alpha}_1 \| \bar{\alpha}_2 \| \cdots \| \bar{\alpha}_n$ be a database encrypted using the assumed FHE scheme, where $\alpha_i = (v_i, w_{i1}, \ldots, w_{i\tau})$ with $v_i, w_{ij} \in \{0,1\}^{\ell-1}$ for $i \in [n]$ and $j \in [\tau]$ and where $\bar{\alpha}_i$ denotes a component-wise encryption of α_i, i.e., $\bar{\alpha}_i = (\bar{v}_i, \bar{w}_{i1}, \ldots, \bar{w}_{i\tau})$. For notational convenience, we will frequently use v_i in place of $\alpha_i.v_i$ and w_{ij} in place of $\alpha_i.w_{ij}$ for $i \in [n]$ and $j \in [\tau]$. The reader may think of D as a collection of n $(\tau+1)$-tuples and $\alpha_i.w_{ij}$ (or w_{ij}) as the value of a key attribute.

For the correctness of our protocols, we assume that $v_i \neq 0$ for all $i \in [n]$. To handle this issue, we employ an FHE encryption scheme with the plaintext space $\mathbb{F}_{2^\ell}$, not $\mathbb{F}_{2^{\ell-1}}$, and assume that each v_i is encoded into $1 \| v_i$ in advance, where $a \| b$ denotes the concatenation of strings a and b. Unless confusion arises, we omit encoding and decoding steps between v_i and $1\|v_i$ in our protocols.

Message encoding into $\mathbb{F}_{2^\ell}$. To encrypt an ℓ-bit message $a = (a_\ell, \ldots, a_1) \in \{0,1\}^\ell$ using an FHE scheme with the plaintext space $\mathbb{F}_{2^\ell}$, we must first encode the message a using $\sum_{i=0}^{\ell-1} a_{i+1} t^i \in \mathbb{F}_{2^\ell}$. We can then write an encryption of the message a as follows:

$$\bar{a} := \mathsf{E}_{pk}\left(\textstyle\sum_{i=0}^{\ell-1} a_{i+1} t^i\right).$$

4.2 Private Query Processing Protocols

The descriptions of our private query processing protocols follow. For completeness, we provide a brief description of a correctness check and an evaluation of the total multiplicative depth for each protocol along with a communication cost analysis on the server side.

Let $J \subseteq [\tau]$ be a set of indices such that $|J| = \rho$.

Conjunctive Queries We first show how to privately process a query with conjunctive conditions. To this end, suppose that a client has an SQL code that requests all $\alpha_i.v_i$'s such that $\alpha_i.w_{ij} = a_j$ for $j \in J$ and $a_j \in \mathcal{P}$. An example of such an SQL code[5] is as follows: for a relation $\mathsf{R}(V, W_1, \ldots, W_\tau)$ and for each encryption $\bar{a}_j = \mathsf{E}(a_j)$,

$$\texttt{select } V \texttt{ from } \mathsf{R} \texttt{ where } W_1 = \bar{a}_1 \texttt{ and } \cdots \texttt{ and } W_\rho = \bar{a}_\rho.$$

Our protocol is described as follows:

[5] It is assumed that a suitable transforming module is present on the client side and that the module takes as inputs a plain SQL code and a set of data (e.g., $a_1, \ldots, a_\rho$) and outputs a transformed SQL code, as in the example given above, by invoking the FHE encryption algorithm. Unless otherwise stated, by an SQL code, we mean a properly transformed SQL code.

1. The client sends an SQL code with the `where` clause correctly encoded in the form of conjunctive combinations of equality, i.e., $\bigwedge_{j\in J}(\alpha_i.w_{ij} = a_j)$.
2. Upon receiving the query, the server parses it and performs the following operations.
   (a) Computes $\bar{\beta}_{ij} = \texttt{EQTest}(\bar{w}_{ij}, \bar{a}_j)$ for $i \in [n]$ and $j \in J$. Note that $\bar{w}_{ij} = \bar{\alpha}_i.\bar{w}_{ij}$.
   (b) Computes $\bar{\gamma}_i = \left(\prod_{j\in J}\bar{\beta}_{ij}\right)\cdot\bar{v}_i$ for $i \in [n]$; hereafter, '$\cdot$' indicates the multiplication of two ciphertexts. Note that $\bar{\alpha}_i.v$ is written as $\bar{v}_i$ for short.
   (c) Sends $\{\bar{\gamma}_1, \bar{\gamma}_2, \ldots, \bar{\gamma}_n\}$ to the client.
3. The client obtains γ_i by decrypting $\bar{\gamma}_i$ for $i \in [n]$.

Correctness. From the properties of the `EQTest` algorithm, $\beta_{ij} = 1$ if $\alpha_i.w_{ij} = a_j$ and $\beta_{ij} = 0$ if $\alpha_i.w_{ij} \neq a_j$. Hence, $\gamma_i = \alpha_i.v_i$ if $\alpha_i.w_{ij} = a_j$ for all $j \in J$; otherwise, $\gamma_i = 0$. Therefore, the client obtains only $\alpha_i.v_i$ such that $\alpha_i.w_{ij} = a_j$ for all $j \in J$. This means that our protocol functions correctly.

Complexity. For Step 2 (b), the protocol requires a multiplicative depth of $\lceil\log(1+\rho)\rceil$, and thus, the total multiplicative depth amounts to $\lceil\log\ell\rceil + \lceil\log(1+\rho)\rceil$. The communication cost on the server side is n ciphertexts using the employed FHE scheme.

Remark 2 *Here, we cannot achieve query privacy that requires hiding from an attacker whose query has been submitted to the server. Query privacy appears to be a stronger requirement than data privacy, ensuring the confidentiality of data returned in response to a submitted query. The current version of our solutions satisfies only the latter data privacy requirement.*

In addition, the server should know the client's public key pk and evaluation key ek to generate ciphertexts and perform evaluation on ciphertexts, respectively. Hence, our solutions in this paper cannot achieve the clients' anonymity requirement.

Disjunctive Queries We next consider a disjunctive query that requests $\alpha_i.v_i$'s such that there is at least one j that satisfies $\alpha_i.w_{ij} = a_j$ among all $j \in J$. The protocol description follows.

1. The client sends an SQL code with a disjunctive `where` condition, i.e., $\bigvee_{j\in J}(\alpha_i.w_{ij} = a_j)$.
2. Upon receiving the request, the server parses it and performs the following operations.
   (a) Computes $\bar{\beta}_{ij} = (\bar{1} - \texttt{EQTest}(\bar{w}_{ij}, \bar{a}_j))$ for $i \in [n]$ and $j \in J$.
   (b) Computes $\bar{\gamma}_i = \left(\bar{1} - \prod_{j\in J}\bar{\beta}_{ij}\right)\cdot\bar{v}_i$ for $i \in [n]$.
   (c) Sends $\{\bar{\gamma}_1, \bar{\gamma}_2, \ldots, \bar{\gamma}_n\}$ to the client.
3. The client obtains γ_i by decrypting $\bar{\gamma}_i$ for $i \in [n]$.

Correctness. By the same argument as above, $\beta_{ij} = 1$ if $\alpha_i.w_{ij} \neq a_j$ and $\beta_{ij} = 0$ if $\alpha_i.w_{ij} = a_j$ using an output of the `EQTest` algorithm. Hence, $\gamma_i = 0$ if $\alpha_i.w_{ij} \neq a_j$ for all $j \in J$, and $\gamma_i = \alpha_i.v_i$ otherwise. Therefore, the client obtains $\alpha_i.v_i$'s such that there is at least one j that satisfies $\alpha_i.w_{ij} = a_j$ among all $j \in J$. Hence, the correctness of the protocol is proven.

Complexity. As in the conjunctive case, the protocol requires a multiplicative depth of $\lceil\log(1+\rho)\rceil$ for Step 2 (b), and thus, the total multiplicative depth amounts to $\lceil\log\ell\rceil + \lceil\log(1+\rho)\rceil$. The communication cost on the server side is also n ciphertexts of the employed FHE scheme.

Threshold Conjunctive Queries Finally, we discuss a protocol for a threshold conjunctive query that requests $\alpha_i.v_i$'s such that the number of j's satisfying $\alpha_i.w_{ij} = a_j$ is greater than T for $j \in J$ and a prefixed threshold $T \in \mathbb{Z}_+$. For the correctness of the proposed protocol, we assume that $\ell > \rho$.

Our protocol for this special type of conjunctive query is as follows.

1. The client sends an SQL code with a threshold conjunctive condition in its `where` clause, i.e., $|\{j \in J : \alpha_i.w_{ij} = a_j\}| > T$.
2. Upon receiving the query, the server parses it and performs the following operations.
   (a) Finds a polynomial $g \in \mathcal{P}[x]$ of $\deg(g) = \rho$ such that $g(t^\kappa) = 1$ for $T < \kappa \le \rho$ and $g(t^\kappa) = 0$ for $0 \le \kappa \le T$.
   (b) Computes $\bar{\beta}_{ij} = \texttt{EQTest}(\bar{w}_{ij}, \bar{a}_j) \cdot \overline{(t+1)} + \bar{1}$ for $i \in [n]$ and $j \in J$.
   (c) Computes $\bar{\zeta}_i = \prod_{j \in J} \bar{\beta}_{ij}$ and then $\bar{\gamma}_i = g(\bar{\zeta}_i) \cdot \bar{v}_i$ for $i \in [n]$.
   (d) Sends $\{\bar{\gamma}_1, \bar{\gamma}_2, \ldots, \bar{\gamma}_n\}$ to the client.
3. The client obtains γ_i by decrypting $\bar{\gamma}_i$ for $i \in [n]$.

Correctness. From the properties of the `EQTest` algorithm over the plaintext domain of characteristic 2, $\beta_{ij} = t$ if $\alpha_i.w_{ij} = a_j$ and $\beta_{ij} = 1$ otherwise. Hence, $\zeta_i = t^\kappa$ for $\kappa = |\{j \in J : \alpha_i.w_{ij} = a_j\}|$. Therefore, if $\kappa > T$, then $g(\zeta_i) = 1$; otherwise, $g(\zeta_i) = 0$ by the definition of the polynomial g. This proves the correctness of our protocol.

Complexity. The protocol requires a multiplicative depth of $\lceil \log \rho \rceil$ to compute $\bar{\zeta}_i$ and a multiplicative depth of $\lceil \log (1+\rho) \rceil$ to evaluate g and then multiply by $\bar{v}_i$. Hence, our protocol requires a total multiplicative depth of $\lceil \log \ell \rceil + 2\lceil \log (1+\rho) \rceil$, and the communication cost to the server is n ciphertexts using the employed FHE scheme.

We generalize the results of our depth analyses from the above sections in Table 2.

Table 2. The multiplicative depth for each query type[†]

Query type	Multiplicative depth[‡]
Conjunction	$\lceil \log \ell \rceil + \lceil \log (1+\rho) \rceil$
Disjunction	$\lceil \log \ell \rceil + \lceil \log (1+\rho) \rceil$
Threshold Conjunction	$\lceil \log \ell \rceil + 2\lceil \log (1+\rho) \rceil$

[†] $\mathcal{M} = \{0,1\}^{\ell-1}$, $\mathcal{P} = \mathbb{F}_{2^\ell}$

[‡] ρ: the number of attributes in the condition

5 A Communication-efficient Improvement

In this section, we devise a method to reduce the communication overhead from n to $\lfloor \delta n \rfloor$ ciphertexts, at the cost of additional computations, when an upper bound δ on the *selectivity* of a database D of n tuples is given.[6]

Following Boneh et al.'s scheme, our strategy takes advantage of polynomial representations. Recall that given a set $S = \{s_1, s_2, \ldots, s_n\}$, we can construct a polynomial representation of S,

[6] The selectivity is defined as the ratio of the number of records that satisfy a given condition with respect to the total number of records in D, and thus, we assume that an upper bound on the selectivity of a database satisfies $0 \le \delta \le 1$.

denoted by $f_S(x)$, as follows:

$$f_S(x) = \prod_{s_i \in S}(x - s_i).$$

Let $\bar{\Gamma} = \{\bar{\gamma}_1, \bar{\gamma}_2, \ldots, \bar{\gamma}_n\}$ be the set of elements that the server obtains at the end of our protocols described in Section 4, and let $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ be the set of plaintexts γ_i corresponding to the ciphertexts $\bar{\gamma}_i \in \bar{\Gamma}$. If we assume that the upper bound on the selectivity of the database D is δ, then there are at most $\lfloor \delta n \rfloor$ ciphertexts of nonzero elements in the set $\bar{\Gamma}$. Without loss of generality, we assume that $\gamma_i = \alpha_i.v_i \neq 0$ for $1 \leq i \leq m$ and that $\gamma_i = 0$ for $m < i \leq n$, where $m \leq \lfloor \delta n \rfloor$.

Next, consider a polynomial representation of Γ:

$$f_\Gamma(x) = \prod_{\gamma_i \in \Gamma}(x - \gamma_i).$$

We see that the polynomial $f_\Gamma(x)$ can be represented by a product of two polynomials $x^{n - \lfloor \delta n \rfloor}$ and $g_\Gamma(x)$, where

$$g_\Gamma(x) = \prod_{1 \leq i \leq \lfloor \delta n \rfloor}(x - \gamma_i),$$

because $\gamma_i = 0$ for all $\lfloor \delta n \rfloor < i \leq n$. Letting $f_\Gamma(x) = \sum_{i=0}^{n} f_i x^i$ and $g_\Gamma(x) = \sum_{i=0}^{\lfloor \delta n \rfloor} g_i x^i$, we can write that $f_{i+(n-\lfloor \delta n \rfloor)} = g_i$ for $0 \leq i \leq \lfloor \delta n \rfloor$ and $f_i = 0$ for $0 \leq i \leq (n - \lfloor \delta n \rfloor)$.

In fact, it is not difficult to reduce the communication overhead using the property described above. For a given set $\bar{\Gamma}$, the server first computes its polynomial representation

$$f_{\bar{\Gamma}}(x) = \prod_{1 \leq i \leq n}(x - \bar{\gamma}_i) = \sum_{i=0}^{n} \bar{f}_i x^i$$

and sends the client the encrypted coefficients $\bar{f}_i$, where $(n - \lfloor \delta n \rfloor) \leq i < n$, rather than the set of all $\bar{\gamma}_i$'s. Note that the server does not need to send $\bar{f}_n$ because f_n is trivially 1. After decrypting the sequence of encrypted coefficients, the client reconstructs a polynomial f_Γ and computes the roots of the polynomial using a root-finding algorithm on an extension field $\mathbb{F}_{2^\ell}$ in [22], whose computational complexity is $\tilde{O}(n\ell^2)$ to find all roots of a degree-n polynomial over a finite field $\mathbb{F}_{p^\ell}$ of small characteristic p. Computing $\bar{f}_i$'s on the server side requires a multiplicative depth of $\lceil \log n \rceil$ in addition.

Estimation with selecting concrete parameters. Here, we compare the communication costs of our protocols under a specific FHE scheme instantiated with concrete parameters. We chose the BGV scheme as an underlying FHE scheme which allows SIMD technique, and set the native message space $\mathcal{M}$ and the plaintext space $\mathcal{P}$ to $\{0,1\}^{31}$ and $\mathbb{F}_{2^{32}}$, respectively. In our estimation, the basic protocols in Section 4 use the BGV scheme instantiated with a multiplicative depth of 20. On the other hand, the protocol in Section 5 uses new instances with multiplicative depths of 30 and 40 for the databases of $n = 2^{10}$ and $n = 2^{20}$ elements, respectively. Specifically, the concrete parameter values used in our computations may be found in reference [23]. Applying SIMD technique, the total ciphertext size is approximately $2 \times (n/32) \times \deg(\Phi) \times \lceil \log Q \rceil$ bits, where the ciphertext space of the lowest level is $\mathbb{Z}_Q[x]/\langle \Phi \rangle$ for a polynomial Φ. We note that $\deg(\Phi)$ is selected as n in Table 2 of [23]. For more details on the parameter selections, see Section 3 in the reference [23].

Table 3 presents a comparison of communication costs with respect to the number of elements n and an upper bound δ on the selectivity. Note that for each case, the parameters in the table allow up to 32767 ($= 2^{15} - 1$) attributes in conjunction and disjunction queries and up to 127 ($= 2^7 - 1$) attributes in threshold conjunction queries. This table demonstrates that our communication efficiency-oriented protocol reduces the communication cost by approximately 88% (resp. 77%) for $n = 2^{10}$ and 80% (resp. 59%) for $n = 2^{20}$ when $\delta = 0.05$ (resp. $\delta = 0.10$).

Table 3. Comparison of Communication Costs[§]

n	δ	Basic Protocol (Section 4) (A)	Improved Protocol (Section 5) (B)	(B)/(A)
2^{10}	0.05	41.42 MB	4.77 MB	11.52 %
	0.10		9.55 MB	23.05 %
2^{20}	0.05	42.42 GB	8.70 GB	20.05 %
	0.10		17.39 GB	40.99 %

[§] $\mathcal{M} = \{0,1\}^{31}$, $\mathcal{P} = \mathbb{F}_{2^{32}}$

Remark 3 *There has been a lot of work to evaluate an upper bound of selectivity on the server side (see references in [24]). However, the server in our setting cannot perform such a selectivity estimation. A rudimentary solution is to make the client measure a bound of selectivity by issuing an aggregate query periodically. Specifically, the client sends to the server a* `count` *query with a proper predicate where the* `count` *function can be easily obtained by a sum of evaluating the predicate on ciphertexts. (In our setting, we may obtain such a protocol by requesting an encrypted sum of* $(\prod \beta_{ij})$*'s (resp.,* $g(\delta_{ij})$*'s) to the server in conjunctive and disjunctive (resp., threshold) protocols. For integer addition evaluation, we can apply the technique in [5] by regarding the plaintext space as* $\{0,1\}$*. It takes a total multiplicative depth of* $\lceil 1 + \log(n-2) \rceil$*, and the communication cost of* $\lceil \log n \rceil$ *ciphertexts.) As a result, the client finds the number of tuples which pass the predicate and the total number of tuples and so he learns a selectivity ratio at this point. Cumulating these ratios will allow the client to predict a more precise selectivity.*

6 Concluding Remarks

We analyzed the multiplicative depth required for private query protocols for application to databases encrypted using an FHE scheme. For this purpose, we first evaluated the depth required for an equality test algorithm with respect to the native message and plaintext spaces of currently available FHE schemes. Our analysis demonstrated that the equality test algorithm consumes the minimum multiplicative depth when the plaintext space of the FHE scheme is a field of characteristic 2. Then, we developed a series of private query processing protocols for conjunctive, disjunctive, and threshold queries on encrypted databases. Furthermore, we analyzed the multiplicative depth and communication cost for each protocol. Finally, we designed a method of reducing the communication overhead from n to $\lfloor \delta n \rfloor$ ciphertexts when we know δ, the upper bound on the selectivity of a database.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. Myungsun Kim was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2014-R1A1A2058377). This work was partially done while visiting Nanyang Technological University, Singapore. Hyung Tae Lee, San Ling, and Huaxiong Wang were supported by Research Grant TL-9014101684-01 and the Singapore Ministry of Education under Research Grant MOE2013-T2-1-041.

References

1. C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Symposium on Theory of Computing Conference (STOC) 2009*, M. Mitzenmacher, Ed. ACM, 2009, pp. 169–178.
2. D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu, "Private database queries using somewhat homomorphic encryption," in *ACNS 2013*, ser. LNCS, M. J. J. Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, Eds., vol. 7954. Springer, 2013, pp. 102–118.
3. L. Kissner and D. X. Song, "Privacy-preserving set operations," in *Advances in Cryptology - CRYPTO 2005*, ser. LNCS, V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 241–257.
4. Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Advances in Cryptology - CRYPTO 2012*, ser. LNCS, R. Safavi-Naini and R. Canetti, Eds., vol. 7417. Springer, 2012, pp. 868–886.
5. J. H. Cheon, M. Kim, and M. Kim, "Search-and-compute on encrypted data," in *Financial Cryptography and Data Security*, 2015, pp. 142–159.
6. C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in Cryptology - CRYPTO 2012*, ser. LNCS, R. Safavi-Naini and R. Canetti, Eds., vol. 7417. Springer, 2012, pp. 850–867.
7. J. Coron, T. Lepoint, and M. Tibouchi, "Scale-invariant fully homomorphic encryption over the integers," in *Public-Key Cryptography - PKC 2014*, ser. LNCS, H. Krawczyk, Ed., vol. 8383. Springer, 2014, pp. 311–328.
8. R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *SOSP 2011*, T. Wobber and P. Druschel, Eds. ACM, 2011, pp. 85–100.
9. S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," *PVLDB*, vol. 6, no. 5, pp. 289–300, 2013.
10. D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *IEEE Symposium on Security and Privacy 2000*. IEEE Computer Society, 2000, pp. 44–55.
11. C. Gentry and S. Halevi, "Implementing Gentry's fully-homomorphic encryption scheme," in *Advances in Cryptology - EUROCRYPT 2011*, ser. LNCS, K. G. Paterson, Ed., vol. 6632. Springer, 2011, pp. 129–148.
12. J. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Advances in Cryptology - CRYPTO 2011*, ser. LNCS, P. Rogaway, Ed., vol. 6841. Springer, 2011, pp. 487–504.
13. Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Foundations of Computer Science (FOCS) 2011*, R. Ostrovsky, Ed. IEEE Computer Society, 2011, pp. 97–106.
14. Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Innovations in Theoretical Computer Science (ITCS) 2012*, S. Goldwasser, Ed. ACM, 2012, pp. 309–325.
15. S. Halevi and V. Shoup, "HElib-An implementation of homomorphic encryption," https://github.com/shaih/HElib/, 2014.
16. M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology - EUROCRYPT 2004*, ser. LNCS, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 1–19.
17. K. B. Frikken, "Privacy-preserving set union," in *Applied Cryptography and Network Security (ACNS) 2007*, ser. LNCS, J. Katz and M. Yung, Eds., vol. 4521. Springer, 2007, pp. 237–252.
18. J. H. Cheon, M. Kim, and K. Lauter, "Homomorphic computation of edit distance," in *Financial Cryptography and Data Security*, 2015, pp. 194–212.
19. G. S. Çetin, Y. Doröz, B. Sunar, and E. Savas, "Depth optimized efficient homomorphic sorting," in *Progress in Cryptology - LATINCRYPT 2015*, ser. LNCS, K. E. Lauter and F. Rodríguez-Henríquez, Eds., vol. 9230. Springer, 2015, pp. 61–80.
20. S. Kaji, T. Maeno, K. Nuida, and Y. Numata, "Polynomial expressions of carries in p-ary arithmetics," *CoRR*, vol. abs/1506.02742, 2015. [Online]. Available: http://arxiv.org/abs/1506.02742
21. J. W. Bos, K. E. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *IMA International Conference on Cryptography and Coding (IMACC) 2013*, ser. LNCS, M. Stam, Ed., vol. 8308. Springer, 2013, pp. 45–64.
22. V. Shoup, "A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic," in *International Symposium on Symbolic and Algebraic Computation (ISSAC) 1991*, S. M. Watt, Ed. ACM, 1991, pp. 14–21.
23. J. van de Pol and N. P. Smart, "Estimating key sizes for high dimensional lattice-based systems," in *IMA International Conference on Cryptography and Coding (IMACC) 2013*, ser. LNCS, M. Stam, Ed., vol. 8308. Springer, 2013, pp. 290–303.
24. V. Markl, N. Megiddo, M. Kutsch, T. M. Tran, P. J. Haas, and U. Srivastava, "Consistently estimating the selectivity of conjuncts of predicates," in *Very Large Data Bases (VLDB) 2005*, K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P. Larson, and B. C. Ooi, Eds. ACM, 2005, pp. 373–384.