Efficient Index-based Search Protocols for Encrypted Databases

Majid Nateghizad, Zekeriya Erkin and Reginald L. Lagendijk
Cyber Security Group, Department of Intelligent Systems, Delft University of Technology, The Netherlands

Keywords: Searching in Encrypted Databases, Homomorphic Encryption, Statistical Query, Indexing, Privacy.

Abstract: It is astonishing to see more and more services built on user-oriented data, providing numerous tools to improve ones daily life. Nowadays, data collected from numerous sources is being used to monitor daily activities, i.e., monitoring patients. These innovations allow for more cost-efficient and scalable solutions. Nevertheless, these types of services can pose a threat to the privacy of individuals due to the possibility of leaking highly privacy-sensitive data. Therefore, it is essential to design such systems in a privacy-preserving manner. Inspired by a real-life project in the health-care domain, we propose to secure the data using encryption, while enabling the involved parties to run queries directly on this encrypted data. A vital component of such a system is searching for specific data entries within a large dataset. In this work, we present two cryptographic protocols that complete such a query by creating an encrypted vector in a simulation secure way. These vectors consist of a 1 for intended database entry, whereas other items would be represented as a 0. By creating index tables before the execution of the queries, it has become possible to execute a search query with high performance. As we show in our analyses, it takes less than one second to find the matching encrypted data-entry within a database with 100K records. Our proposal is generic, can be applied to several application domains, and practically compared to similar works.

1 INTRODUCTION

A real-life problem motivates the work presented in this paper: A hospital wants to monitor its patients using off-the-shelf smart devices (Treskes et al., 2017) that measure, among other things, a patient's weight, ECG, blood pressure, and blood sugar level. These devices connect to the smartphone of a patient, who needs to be monitored on a daily basis. A mobile application then sends the measurements to the central server of the vendor. Afterward, the hospital can use a web-based application to check the measurements for any particular patient. The primary reasons to use such a system are straight-forward: scalability and cost reduction (Goldschmidt, 2005).

Unfortunately, the whole system relies on the assumption that the vendor is trust-worthy and it has a secure method to protect against both internal and external attackers. All recent incidents show that this is not yet the case (Meingast et al., 2006). Currently, it is possible for a hacker to break into the data servers and steal privacy-sensitive medical data, such an attacker can either be a malicious employee or someone who makes a genuine mistake (Johnson, 2009). It is essential to propose a system where sensitive data are protected while enabling medical institutions to monitor their patients remotely.

In order to create a secure system with this layout, we propose to encrypt measurements directly on the smartphone of the user before sending them to the vendor. We aim to encrypt the data using a homomorphic encryption scheme which enables data processing while encrypted, without revealing the content of data to the vendor or any third party. More precisely, what is needed for the above system is to identify a patient, or a group of patients, with specific conditions, e.g., people with high blood pressure within a particular time period. Provided that we require semantic security, it is challenging to find all the data for given conditions, since it requires 'searching through the encrypted database.'

Searching in encrypted databases has been a challenge for researchers for many decades. Proposed solutions vary in the cryptographic tools used for encrypting data. Examples include: schemes built on attribute-based encryption (Bethencourt et al., 2007), homomorphic encryption (Chung et al., 2010) and special constructions such as Oblivious RAMS (Ostrovsky, 1990; Goldreich and Ostrovsky, 1996). The focus has been on improving efficiency; the current state-of-the-art is not as practical as searching within a plaintext database where different techniques can be

Nateghizad, M., Erkin, Z. and Lagendijk, R.
Efficient Index-based Search Protocols for Encrypted Databases.
DOI: 10.5220/0006869802700281
In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications (ICETE 2018) - Volume 2: SECRYPT*, pages 270-281
ISBN: 978-989-758-319-3

used to speed up the search function (e.g., creating hash tables). Therefore, there is a need for further research to achieve higher efficiency.

Inspired by the medical application, in this work, we assume that patients have a smartphone, which can collect measurements from one or more smart devices. The data is then sent to the vendor's storage unit, which can either be a local or cloud-based database. More interestingly, a patient might utilize different smart devices from different vendors. The hospital that wants to monitor a specific patient, or a group of patients, should be able to retrieve the related data without leaking information to any of the vendors. This application setting is challenging for three reasons: 1) we want to enable hospitals to retrieve data on sophisticated queries, 2) we also assume multiple devices from different vendors and 3) the amount of data collected from the patients is significantly large. To achieve our goal, we need to identify the data entries, which are all encrypted, that match the query provided by the hospital. More precisely, we assume that there is a global (virtual) database with encrypted entries from all devices. Given that database, we want to create an encrypted, binary vector such that a vector element is 1 for corresponding database entry and 0, for all other tuples. Given such an encrypted binary vector, it is possible to build numerous services such as i) generating statistics (i.e., counting, averaging), ii) data aggregation, and iii) private data retrieval.

To obtain such a vector, we present two cryptographic protocols for secure searching, IBSvI and IBSvII. In IBSvI, we propose a computation-wise efficient searching protocol. In IBSvII, the computation of generating the encrypted binary vector is performed in one party. However, IBSvII introduces more computational overhead. These protocols rely on creating index tables and updating them with each input received from a device. The index tables are then used later to execute queries and find specific database entries at significantly lower cost. Our proposal has several advantages over existing works: 1) our protocols are designed for numerical data, in contrast to current work that relies on exact match, 2) our proposal supports conjunction queries with "AND", 3) our proposal is simulation secure; it leaks no private information including search pattern and access pattern to the involved entities, and 4) our protocols enable generating statistics from encrypted data based on the given conditions.

2 RELATED WORK

Ostrovsky and Goldreich (Ostrovsky, 1990; Goldreich and Ostrovsky, 1996) introduced ORAM where it is possible to evaluate any query, while the access pattern is kept hidden. ORAM lets users upload their private data to a remote storage in encrypted form, and still have random access to their data in a secure way. However, ORAM allows users to access only one entry at a time with a logarithmic number of communication rounds for each read. Moreover, in ORAM, users should know the location of the data that they are looking for in the database. Later works (Song et al., 2000; Goh, 2003; Chang and Mitzenmacher, 2005) proposed more efficient ways of searching by using weaker security models. Song et al. (Song et al., 2000) introduced a private key based searching that is communication-wise more efficient than ORAM. The secure searching in (Song et al., 2000) is based on generating and storing a two-layer ciphertext in the remote storage unit. Although the introduced encryption scheme by Song et al. is proven to be secure, the searching procedure reveals the access pattern. Similarly, Stefanov et al. (Stefanov et al., 2014) combined a secure search and ORAM, where the keywords are kept confidential, but the search protocol still reveals the access pattern to the remote storage. To improve the searching performance, as one of the limitations of Song et al. (Song et al., 2000), Goh (Goh, 2003), and Chang and Mitzenmacher (Chang and Mitzenmacher, 2005) proposed two new secure searching protocols with indexing. They constructed an index table alongside each set of data in the remote storage. The remote storage uses the index tables to find the matching data instead of checking every single encrypted data. Although (Goh, 2003; Chang and Mitzenmacher, 2005) are efficient, their proposals leak access pattern to the untrusted parties.

Curtmola et al. (Curtmola et al., 2011) also presented a semantically secure search by using asymmetric data encryption, which is capable of finding desired data in sub-linear search time. In (Curtmola et al., 2011), each user constructs an index, which includes every possible data that can appear in a query, then the index table is deterministically encrypted and outsourced to the remote storage. To perform a search, a user constructs a query that contains a token that is a deterministic function of the search data and sends the token to the remote storage. Then, the remote storage unit searches for the specific data in each set. Although the proposed searching technique is fast, it has two limitations: 1) users cannot update the index table of their data unless they generate the index table again, and 2) the searching technique still reveals the access pattern. To overcome the challenge of updating index table in (Curtmola et al., 2011), Kamara et al. (Kamara et al., 2012) introduced an improvement, which enables updating the index table. However, the problem of revealing the access pattern is not addressed in that work. (Ding et al., 2017) proposes an efficient and secure search that supports top-k similarity search over encrypted data by using a random traversal algorithm. However, in (Ding et al., 2017), users cannot evaluate their queries, but only the data owner. (Miyoshi et al., 2017) presents an efficient search technique over encrypted data that uses Bloom filter as the indexing technique. Although using Bloom filter introduces false-positive results, it makes the size of the indices very small and independent from the security parameter at the cost of leaking number of matches.

Boneh et al. introduced the first Public-key Encryption with Keyword Search (PEKS) (Boneh et al., 2004), which was shown later that it leaks user's access pattern. Furthermore, (Boneh et al., 2004) is insecure against offline keyword guessing attack (Byun et al., 2006). Boneh et al. proposed another PEKS (Boneh et al., 2007), which is based on PIR (Chor et al., 1995) and Bloom filters, where the aim is to hide the access pattern. Although (Boneh et al., 2007) is secure, PIR-based schemes are computationally expensive. Moreover, in (Boneh et al., 2007) the number of matches that can be found in the remote storage is fixed beforehand to not leak the number of matches. To reduce the search overhead, Bellare et al. (Bellare et al., 2007) introduced an efficient public-key searchable encryption (ESE), which achieved an optimal search time. In contrast to PEKS, ESE allows other users to generate tokens and search for data in the remote storage unit only by having the public key. However, ESE encryption scheme is deterministic and vulnerable to brute-force attacks.

Sahai and Waters (Sahai and Waters, 2005) introduced a new encryption scheme called Attribute-Based Encryption (ABE), which is capable of using an arbitrary string as the public key. In ABE, a ciphertext is not generated for a particular receiver, but for whom possess the desired attributes. In later works, Goyal et al. (Goyal et al., 2006) and Bethencourt et al. (Bethencourt et al., 2007) revised the ABE and introduced Key-Policy Attribute-Based Encryption (KP-ABE) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE). Then, Han et al. (Han et al., 2014) proposed a new encryption scheme, Attribute-Based Encryption with Keyword Search (ABEKS), which enables a multi-user access control based on KP-ABE. In ABEKS, a token is generated by using user's private key, and it consists of the desired data to be searched in the remote storage unit. However, in (Han et al., 2014) search is realized through decryption of ciphertexts in the database, which introduces a significant computational overhead. Moreover, this technique requires generating a trapdoor, which necessitates collaboration with data owner. (Zhang et al., 2016) proposes a privacy-preserving ranked multi-keyword search in a multi-owner model. This approach allows each data owner to use his private key for the encryption. However, this proposal suffers from the high computational cost of searching. Guo et al. (Guo et al., 2017) proposed a secure search that supports multiple data owners setting. (Guo et al., 2017) also enables rank search based on the relevance of documents and keyword, and quality of documents. They also propose an efficient indexing structure, group keyword balanced binary tree (GBB tree), to achieve higher efficiency in searching. However, in (Guo et al., 2017), a trusted third proxy is used to facilitate data outsourcing and query evaluation. Moreover, the improvement over BB-tree leaks private information regarding access pattern. As it is stated in (Guo et al., 2017), the GBB-tree may not access one or multiple subtrees to reduce computation overhead, which can reveal access pattern.

Chung et al. (Chung et al., 2010) introduced a secure outsourcing protocol based on Gentry's fully homomorphic encryption scheme (Gentry, 2009). Although in (Gentry, 2009) confidentiality of data is preserved, while data processing remains possible, its computational overhead is still a challenge. Li et al. (Li et al., 2012) proposed a method to apply homomorphic encryption with an overhead linear in the number of the records. Xiong et al. (Xiong et al., 2013) introduced a ciphertext-policy-ABE (CP-ABE) searchable encryption by using homomorphic encryption, where its search time is proportional to the size of the dataset. Bösch et al. (Bösch et al., 2012) proposed a scheme, BTH$^+$, which is a combination of somewhat homomorphic encryption, and indexing technique of Chang and Mitzenmacher (Chang and Mitzenmacher, 2005). In Bösch's work (Bösch et al., 2012), only the data owner can perform a search over the data, since generating trapdoor requires the possession of the private key.

Gentry et al. (Gentry et al., 2015) proposed a secure searching based on ORAM and Somewhat Homomorphic Encryption (SHE) scheme. Although using ORAM in (Gentry et al., 2015) prevent information leakage in searching, it is communication-wise expensive. Popa *et al.* (Popa et al., 2012) introduced one of the most well-known solutions, CryptDB, for searching over encrypted data. CryptDB uses different encryption schemes depending on the type of the given query to be evaluated over encrypted data. However, CryptDB leaks the number of matches to the untrusted server. Moreover, in (Akin and Sunar, 2015), the authors show that CryptDB is insecure because it does not provide integrity for the query. Krell et al (Krell et al., 2017) introduce another secure searching using ORAM, SHE, and Bloom filter that is significantly more efficient. However, they achieve such efficiency at the cost of leaking access pattern. Moreover, the work in (Krell et al., 2017) suffers from high storage complexity, where 100K of records each having four searchable keywords results in an encrypted index that using 75GB of RAM. Another drawback of (Krell et al., 2017) is that it requires the data owner to be online for searching. Table 1 summarizes the performance and security of the state-of-the-art searching techniques and our secure searching protocols. In Table 1, we denote multi-reader and multi-writer setting as M-M, n is the total number of records, n_v^a is the number of records having the attribute a equal to v, d is the number of data owners, λ is a security parameter, and k stands for top-k documents as described in (Guo et al., 2017). Note that, in contrast to the existing works, our proposal only addresses the problem of finding the matching records, not retrieving them. Thus, the current works and ours are not comparable concerning performance, communication/computation-wise.

3 SECURE SEARCHING PROTOCOLS

In this work, there are five parties: 1) users, 2) data storage units, 3) query issuer, 4) remote computation system, and 5) key manager:

i) Key Manager (KM): KM generates a pair of public and private keys and shares the public key with the other parties. KM also collaborates with RCS to perform two-party computations such as secure decryption.

ii) Users: They are the owners of private data (patients) that are stored in remote data storage in encrypted form. The users send their measurements to the corresponding vendors. The data is consisting of several attributes, denoted by p_i. Therefore, the users' data structure is a tuple $T(id_u, p_i, id_{p_i}, v)$, where id_u is the user identity, p_i is an attribute (e.g., date, time, age, device id, etc), id_{p_i} is an unique identity for p_i, and v is the measurement. The users send their encrypted tuples to their data storage units.

iii) Data Storage Unit (DSU): Each data storage unit collects data from one or multiple users and sends the data to a remote computation system, cloud, on a regular basis. In our scenario, DSUs are the vendors, who are offering smart devices to people.

iv) Query Issuer (QI): QI is interested in processing users' data (i.e., hospital or medical research institutes). In our work, QI can ask for generating statistics like counting, averaging, and data aggregating. QI constructs a query that includes one or multiple attributes as $q : \{Q_T, \acute{p_i}, \acute{id}_{p_i}\}$ based on the type of result that QI is interested in. Q_T defines the type of the query like counting, each $\acute{p_i}$ represents the value of an attribute. To prevent private data leakage, QI encrypts p_i values. id_{p_i} are meta-data that describe $\acute{p_i}$ referring to what type of attribute (i.e., blood pressure or heart beating rate).

v) Remote Computation System (RCS): RCS is a considered to be a cloud storage and processing unit that has sufficient computational and storage capacity. RCS receives and stores the encrypted data from all DSUs. RCS also receives encrypted queries from QI. Henceforward, we denote the j^{th} tuple in RCS as T_j.

For our constructions, we use an additively homomorphic and semantically secure encryption scheme, namely Paillier (Paillier, 1999), and a fully homomorphic scheme, Fan-Vercanteren (FV) (Fan and Vercauteren, 2012). Our system is designed under the assumption of semi-honest security model (Goldreich, 2004). In our setting, the aim is to protect private data of users and QI from both KM and RCS, which are semi-trusted, during the evaluation of the query q provided by QI. The results of queries are kept hidden from KM and RCS.

3.1 IBSvI

Searching in IBSvI consists of two phases: First, we generate indices, which is done before the execution of the protocol (off-line phase) and second, we invoke the searching protocol upon receiving the query q from QI (on-line phase).

3.1.1 Generating Indices

Indices are constructed in four steps:

1. Users change the measurements of each attribute $p_z, z \in \{0, \cdots, \alpha - 1\}$ to binary form $(p_z)_i$, where $i \in \{0, \cdots, e \leq \ell - 1\}$ and e is the bit-length of p_z. Then, they assign zero to the rest of bits from $(p_z)_e$ to $(p_z)_{\ell-1}$.

Table 1: Summary of schemes. Communication round is denoted as CR, data transmission as DT, and statistical query as SQ.

Scheme	Search	CR	DT	Leakage	SQ	"AND"	M-M
(Goldreich and Ostrovsky, 1996)	$O(n\log^2 n)$	$O(\log n)$	$O(\log^2 n)$	no leakage	no	no	no
(Krell et al., 2017)	$O(\log n)$	$O(\log^2 n)$	$O(\log^3 n)$	access pattern	no	yes	no
(Gentry et al., 2015)	$O(\log n)$	$O(\log n)$	$O(n\log^2 n)$	no leakage	no	yes	no
(Guo et al., 2017)	$O(n)$	$O(d)$	$O(k)$	access pattern	no	yes	yes
(Miyoshi et al., 2017)	$O(\log 2^m + n_v)$	$O(1)$	$O(n_v)$	access pattern	no	no	no
(Kamara et al., 2012)	$O(n_v)$	$O(1)$	$O(\lambda)$	access pattern	no	no	no
IBSvI	$O(mn)$	$O(1)$	$O(n)$	no leakage	yes	yes	yes
IBSvII	$O(mn)$	$O(1)$	$O(n)$	no leakage	yes	yes	yes

Table 2: List of symbols.

Symbol	Description	Symbol	Description
pk/sk	public/secret key	n	encryption modulus
$\mathcal{E}_{pk}$	encryption	$\mathcal{D}_{sk}$	decryption
κ	security parameter	r, r_z, θ	random number in $\mathcal{Z}_n$
T	tuple of multiple attributes	p	attribute
α	number of attributes in a query	$bl(p)$	bit length of p
ℓ	maximum bit-length of attributes	φ	false positive rate control
Ans	result of query evaluation	$\acute{\rho}$	package capacity in Paillier
Q_T	query type	id_{p_i}	meta data for p_i
UnPerm	inverse of Perm	$[x]$	$\mathcal{E}_{pk}(x)$
$\hat{\rho}$	number of item can be packed in FV	index^{p_i}	index tables for p_i
index$^{p_i}_{i,j}$	i^{th} column and j^{th} row in index^{p_i}	$d(a,b)$	Hamming distance of a and b
$T^{p_i}_j$	value of p_i in j^{th} record	$(T^{p_i}_j)_i$	i^{th} least significant bit of $T^{p_i}_j$
$(p_i)_j$	j^{th} least significant bit of p_i	ω	total number of records
Perm	permutation function	Max$_\alpha$	total number of attributes

2. Users multiply each $(p_z)_i$ by 2^i, $i \in \{0, \cdots, e\}$, encrypts the results as a tuple $T^{p_z} :< id_{p_z}, [id_u], [(p_z)_i] >$ and send them to their DSUs, who send the encryptions to RCS later.
3. RCS creates an index for each possible attribute (there are Max$_\alpha$ attributes in total), where each index has ω rows that is the total number of tuples collected from DSUs and ℓ columns.
4. RCS locates each encrypted tuple $[(T^{p_z})_{i,j}]$ (i^{th} bit of the j^{th} tuple received from DSUs) in the corresponding generated index, $index^{[p_z]}_{i,j} \leftarrow [(T^{p_z})_{i,j}]$, $i \in \{0, \cdots, \ell-1\}$.

We present values in the binary form to check whether each encrypted data stored by RCS matches a particular encrypted data in q without invoking any two-party protocol. Each value is multiplied by powers of 2 to eliminate false positive results in our construction. Protocol 1 shows the steps that RCS takes to generate indices. It indicates that Max$_\alpha$ indices are created, one for each attribute. According to Protocol 1, it is explicit generating indices in RCS is computation-free and storage-demanding.

Protocol 1: RCS:GenIndex.

```
Require: $[T^{p_z}_j]$
Ensure: indices for each $p_z$
1: for $z = 0$ to Max$_\alpha - 1$ do
2:     Create index$^{p_z}$
3:     for $i = 0$ to $\ell - 1$ do
4:         for $j = 0$ to $\omega - 1$ do
5:             index$^{[p_z]}_{i,j} \leftarrow [(T^{p_z})_{i,j}]$
6:         end for
7:     end for
8: end for
```

3.1.2 Secure Searching

Protocol 2 shows the process of evaluating an encrypted query $[q]$ over the encrypted databases using the generated indices.

1. $QI^I_{f_1}$: Once RCS generates the indices, QI constructs a query q including $e \leq \alpha$ attributes $\acute{p}_z$, $z \in \{0, \cdots, e-1\}$, which QI is interested in. Similar to the process of index generation, QI converts the values of its attributes to binary form. Then it

Protocol 2: IBSvI.

```
Require: $\text{index}^{p_z}$
Ensure: Find matches
1:  $e \in \{0, \cdots, u \le \alpha - 1\}$                                   $QI^I_{f_1}$
2:  $[q] \leftarrow \{Q_T, [-2^i(\acute{p_e})_i], id_{\acute{p_e}}\}$
3:  for $z = 0$ to $\alpha - 1$ do
4:      for $i = 0$ to $\ell - 1$ do
5:          for $j = 0$ to $\omega - 1$ do
6:              $r \xleftarrow{\$} \{0, 1\}$
7:              $\text{index}^{p_z}_{i,j} \leftarrow \text{index}^{p_z}_{i,j} \oplus (r * 2^i)$          $RCS^I_{f_1}$
8:              $(\acute{p_z})_i \leftarrow (\acute{p_z})_i \oplus (-r * 2^i)$
9:              $[d^{p_z}_{i,j}] \leftarrow (\text{index}^{p_z}_{i,j} + [(\acute{p_z})_i])$
10:         end for
11:     end for
12: end for
13: for $j = 0$ to $\omega - 1$ do
14:     for $z = 0$ to $\alpha - 1$ do
15:         $[R^{p_z}_j] \leftarrow [0]$
16:         for $i = 0$ to $\ell - 1$ do
17:             $[R^{p_z}_j] \leftarrow [R^{p_z}_j + d^{p_z}_{i,j}]$          $RCS^I_{f_2}$
18:         end for
19:     end for
20:     $r_z \xleftarrow{\$} \{2^{\varphi-1}, \cdots, 2^{\varphi} - 1\}$
21:     $\theta \xleftarrow{\$} \{1, \cdots, Max_\alpha - \alpha + 1\}$          $RCS^I_{f_3}$
22:     $[y_j] \leftarrow [\sum_{z=0}^{\alpha-1}(R^{p_z}_j * \theta r_z)]$
23: end for
24: $\acute{y_j} \xleftarrow{\$} \{0, \cdots, 2^{\ell-1}\}$
25: $\acute{r} \xleftarrow{\$} \{0, \cdots, \omega - 1\}$          $RCS^I_{f_4}$
26: InsertZero($\acute{y}, \acute{r}$)
27: $[y] \leftarrow [y] || [\acute{y}]$
28: Perm($[y_j]$), $j \in \{0, \cdots, 2\omega - 2\}$          $RCS^I_{f_5}$
29: $y_j \leftarrow Dec_{key}([y_j])$
30: $I_j \leftarrow (y_j = 0)?1 : 0$          $KM^I_{f_1}$
31: $[I_j] \leftarrow Enc(Ans_j)$
32: $[I_j] \leftarrow$Reverse-Perm($[I_j]$)          $RCS^I_{f_6}$
33: $[Ans_j] \leftarrow$ Half($[I_j]$) : $j \in \{0, \cdots, \omega - 1\}$
```

computes $(\acute{p_z})_i \leftarrow (\acute{p_z})_i * (-2^i)$, $i \in \{0, \cdots, e-1\}$ and assigns zero to $\{(\acute{p_z})_e, \cdots, (\acute{p_z})_{\ell-1}\}$.

2. $RCS^I_{f_1}$: RCS tosses a random coin r for each pair of $(\text{index}^{p_z}_{i,j}, (\acute{p_z})_i)$ to compute $(\text{index}^{p_z}_{i,j} \oplus (r * 2^i), (\acute{p_z})_i \oplus (-r * 2^i))$. Note that the values of $\text{index}^{p_z}_{i,j}$ is in encrypted form. Thus, to compute $\text{index}^{p_z}_{i,j} \oplus (r * 2^i)$, RCS checks whether $r = 0$, $\text{index}^{[p_z]}_{i,j} \leftarrow (r = 0)?(\text{index}^{[p_z]}_{i,j}) : ([2^i] - \text{index}^{[p_z]}_{i,j})$. Similarly, to compute $(\acute{p_z})_i \oplus (-r * 2^i)$, RCS performs $[(\acute{p_z})_i] \leftarrow (r = 0)?[(\acute{p_z})_i] : (-([2^i] + [\acute{p_z})_i])$. Afterwards, RCS computes $[d^{p_z}_{i,j}]$, which is clearly zero if $|\text{index}^{p_z}_{i,j}| = |(\acute{p_z})_i|$ that means the i^{th} bit of $\text{index}^{p_z}_j$ matches the i^{th} bit of $\acute{p_z}$ in q.
3. $RCS^I_{f_2}$: To check if $\acute{p_z}$ equals p_z of the j^{th} tuple in the database, RCS checks whether the Hamming distance d between them is zero, $d(\text{index}^{p_z}_j, \acute{p_z}) = 0$. To obtain d, RCS computes $R^{p_z}_j \leftarrow \sum_{i=0}^{\ell-1}[d^{p_z}_{i,j}]$.
4. $RCS^I_{f_3}$: In case of there are multiple $\acute{p_z}$ in query q, RCS checks if the all the $\acute{p_z}$ matches the corresponding $[p_z]$ in each tuple. To do so, RCS can simply compute $[y_j] \leftarrow \sum_{z=0}^{\alpha-1}[R^{p_z}_j]$, if $[y_j = 0]$ then it means the j^{th} tuple in RCS matches $[q]$. However, there is a possibility of obtaining false positive result. Thus, RCS chooses α uniformly distributed random number r_z, $z \in \{0, \cdots, \alpha - 1\}$ and another random number θ. r_z values are used in the protocol to decrease the false positive rate and θ is used for the security reasons.
5. $RCS^I_{f_4}$: In this step RCS choose ω uniformly random numbers $\acute{y_j} \in \{0, \cdots, 2^{\ell-1}\}$ and another random number $\acute{r}$. In this step, RCS inserts ω random numbers, which has $\acute{r}$ zeros, to $[y]$. This step prevent the KM to learn about the number of matches and other statistical information from y_j values.
6. $RCS^I_{f_5}$: RCS permutes $[y_j]$ to not let the KM learn about the locations of tuples that matched q.
7. $KM^I_{f_1}$: The KM decrypts given $[y_j]$, $j \in \{0, \cdots, 2\omega - 2\}$, and checks whether $y_j = 0$. Then, it creates an array I_j, filled with binary values such that $I_j \leftarrow (y_j = 0)?1 : 0$.
8. $RCS^I_{f_6}$: RCS reverse permutes $[I_j]$ and removes the dummy encryptions added to $[y_j]$ in RCS_{f_4}.

Finally, RCS has an encrypted binary array, which represents the location of matching tuples in the database. To optimize IBSvI regarding computation and communication, we also apply data packing (Erkin et al., 2012; Troncoso-Pastoriza et al., 2007) on our protocol. Note that we cannot apply data packing on $[I_j]$ because it prevents performing reverse permutation and removing dummy encryptions in $RCS^I_{f_6}$. However, we can skip $RCS^I_{f_6}$, and perform $RCS^I_{f_4}$ and $RCS^I_{f_5}$ on the database itself, where $\acute{j} = \acute{r} = 0$, for evaluation of Q_T. This modification allows to pack $[I_j]$ and reduces both communication and computation cost.

3.2 IBSvII

In IBSvII, we achieve a communication cost-free searching algorithm. Similar to IBSvI, there are two phases in IBSvII, indexing and searching. The index tables generation phase is identical to IBSvI except we do not multiply $(p_z)_i$ by 2^i.

Protocol 3 shows how IBSvII works. There are four steps in IBSvII, $QI^{II}_{f_1}$, $RCS^{II}_{f_1}$, $RCS^{II}_{f_2}$, and $RCS^{II}_{f_3}$.

1. $QI^{II}_{f_1}$: Similar to $QI^I_{f_1}$, QI generates the query with $e \le \alpha$ attributes p_z, $z \in \{0, \cdots e-1\}$. Then, QI represents the the values of the attributes to binary form, encrypts them, and send $\{[1 -$

Protocol 3: IBSvII.

```
Require: index$^{p_z}$
Ensure: Find matches
 1: $e \in \{0, \cdots, u \le \alpha - 1\}$                              $QI^{II}_{f_1}$
 2: $[q] \leftarrow \{Q_T, [-(\acute{p_e})_i], id_{\acute{p_e}}\}$
 3: for $z = 0$ to $\alpha - 1$ do
 4:     for $i = 0$ to $\ell - 1$ do
 5:         for $j = 0$ to $\omega - 1$ do
 6:             $[d^{p_z}_{i,j}] \leftarrow (\text{index}^{p_z}_{i,j} + [(\acute{p_z})_i])^2$      $RCS^{II}_{f_1}$
 7:         end for
 8:     end for
 9: end for
10: for $j = 0$ to $\omega - 1$ do
11:     for $z = 0$ to $\alpha - 1$ do
12:         $[R^{p_z}_j] \leftarrow [1]$
13:         for $i = 0$ to $\ell - 1$ do
14:             $[R^{p_z}_j] \leftarrow [R^{p_z}_j \times d^{p_z}_{i,j}]$              $RCS^{II}_{f_2}$
15:         end for
16:     end for
17:     $[Ans_j] \leftarrow [\prod_{z=0}^{\alpha-1} R^{p_z}_j]$                  $RCS^{II}_{f_3}$
18: end for
```

$(p_z)_0], \cdots, [1-(p_z)_{\ell-1}]\}$ to RCS. Afterward, QI sends Q_T plus necessary meta-data to RCS.

2. $RCS^{II}_{f_1}$: This step computes $[d^{p_z}_{i,j}]$ like in $RCS^{I}_{f_1}$, however, $[d^{p_z}_{i,j}]$ are binary values. The challenge in computing $[d^{p_z}_{i,j}]$ in $RCS^{I}_{f_1}$ was that it could be a combination of positive and negative numbers such that their addition $R^{p_z}_j$ becomes zero, which is considered as a false positive result. In $RCS^{II}_{f_1}$, each $[d^{p_z}_{i,j}]$ is squared, which solves the problem of negative numbers. Recall that $(\text{index}^{p_z}_{i,j} + (\acute{p_z})_i) \in \{-1,0,1\}$, thus $d^{p_z}_{i,j} \in \{0,1\}$ in $RCS^{II}_{f_1}$.
3. $RCS^{II}_{f_2}$: After computation of $[d^{p_z}_{i,j}]$, we compute $R^{p_z}_j \leftarrow R^{p_z}_j \times d^{p_z}_{i,j}$. $R^{p_z}_j$ remains one if the $\acute{p_z}$ in the query q matches p_z of j^{th} record in the database.
4. $RCS^{II}_{f_3}$: This step apply the "AND" connections among multiple $R^{p_z}_j$ by computing $\prod_{z=0}^{\alpha-1} R^{p_z}_j$ to obtain Ans_j.

In IBSvII, we can use batching to reduce the computational overhead. Batching enables not only the addition of two packed ciphertexts but also supports multiplication. Thus, all the operations stated in Protocol 3 can be performed over packed ciphertexts without collaboration with KM.

4 SECURITY ANALYSES

We consider the semi-honest security model (Goldreich, 2004), where parties are assumed to be honest in following the protocol description, while they are curious to obtain more information than they are entitled to. Given that the only RCS gets is the encrypted output from the protocol, KM should not be able to distinguish if RCS has a different input and RCS should not learn more information than the output of the protocol. We also assume that parties do not collude with each other.

4.1 Security of IBSvI

Let $RCS^I_f = (RCS^I_{f_1}, \cdots, RCS^I_{f_6})$, $KM^I_f = (KM^I_{f_1})$, and $f = (RCS^I_f, KM^I_f)$ to be the PPT functionality for IBSvI. The view of the ith party ($i \in \{RCS, KM\}$) during the execution of IBSvI on $(index^{p_z}, \phi)$ and security parameter n is denoted by $view^{IBSvI}_i(index^{p_z}, \phi, n) = (w, r^i; m^i_1, \cdots, m^i_t)$, where $w \in \{index^{p_z}, \phi\}$ based on the values of i, r^i are the ith party internal random numbers, and m^i_j represents the jth message that is received by ith party. Note that KM does not have any initial input, thus its input is denoted as ϕ. $output^{IBSvI}_i(index^{p_z}, \phi, n)$ represents the output of each party during the execution of IBSvI. To represent the joint output of both parties, we denote

$$output^{IBSvI} = (output^{IBSvI}_1(index^{p_z}, \phi, n), \\ output^{IBSvI}_2(index^{p_z}, \phi, n)). \tag{1}$$

Definition 4.1. *It can be proven that IBSvI securely computes $f = (RCS^I_f, KM^I_f)$ in the semi-honest security setting if there exits PPT algorithms Sim_{RCS} and Sim_{KM} such that:*

$$\{(Sim_{RCS}(1^n, index^{p_z}, RCS^I_f, f))\} \overset{c}{\equiv} \{(view^f_{RCS} \\ (index^{p_z}, \phi, n), output^f(index^{p_z}, \phi, n))\} \tag{2}$$

and

$$\{(Sim_{KM}(1^n, \phi, KM^I_f, f))\} \overset{c}{\equiv} \{(view^f_{KM} \\ (index^{p_z}, \phi, n), output^f(index^{p_z}, \phi, n))\} \tag{3}$$

Theorem 1. *The protocol IBSvI is simulation secure and securely computes the functionality f, when the party RCS is corrupted by adversary $\mathcal{A}_{RCS}$ in the presence of semi-honest adversaries.*

Proof. We need to show that RCS cannot computationally distinguish between generated messages and outputs from $\mathcal{S}_2$ and $\mathcal{S}_3$, and randomly generated data. RCS receives an output from $\mathcal{S}_2$, $[\widehat{Ans}_j]$, and a message from $\mathcal{S}_3$, $[(\hat{p}_z)_i]$. Protocol 6 shows the simulators for KM and QI that are $\mathcal{S}_2$ and $\mathcal{S}_3$, respectively. Note that because of the space limitation, only the message from $\mathcal{S}_3$ to RCS, $[\hat{p}_z)_i]$, is presented in Protocol

6. Given $index^{p_z}$ and 1^n (security parameters), RCS works as follow:

1. RCS chooses three uniformly random tapes r, r_z, and $\acute{r}$ for RCS_f.
2. $\mathcal{S}_3$ randomly generates u $\hat{p}_z \in \mathbb{N}$, where $u \leq \alpha$, u random meta data, and a random $\hat{Q_T}$. Then, $\mathcal{S}_2$ forms $[q] \leftarrow \{\hat{Q}_T, [(\hat{p}_z)_i]||\hat{MD}_i\}$, where $i \in \{0,\cdots,u \leq \alpha-1\}$, ans send $[q]$ to RCS.
3. RCS executes $RCS^I_{f_i}, i \in \{1,\cdots,5\}$, and it outputs $[\hat{y}_j]$ to $\mathcal{S}_2$.
4. $\mathcal{S}_2$ tosses j coins $\hat{I}_j$ and sends $[\hat{I}_j]$ to RCS.
5. RCS performs UnPerm$[\hat{I}_j]$, and outputs $[\hat{Ans}_j] \leftarrow$Half$([\hat{I}_j])$

The output of the simulation can be written as: $Sim_{RCS}(1^n, index^{p_z}, RCS^I_f, f) = (index^{p_z}, r, r_z, \acute{r}; [\hat{I}_j], [(\hat{p}_z)_i]; ([\hat{Ans}_j], \phi)])$.
The real view of RCS can be presented as $view^f_{RCS}(index^{p_z}, [(\acute{p_z})_i, n]) = (index^{p_z}, r, r_z, \acute{r}; [I_j], [(\acute{p_z})_i])$. And the output of the real view is $output^f(index^{p_z}, [(\acute{p_z}])_i) = ([Ans_j], \phi)$. It can be observed that the encryption pairs $([\hat{I}_j], [(\hat{p}_z)_i]))$ and $([I_j], [(\acute{p_z})_i])$ are indistinguishable, since the crypto-scheme used in IBSvI is semantically secure. For the same reason $\mathcal{A}_{RCS}$ cannot distinguish between $[Ans_j]$ and $[\hat{Ans}_j]$. Recalling that RCS is also given meta-data that describes the query type and attributes in q, RCS cannot see if the provided meta-data are corresponding to the attributes $\hat{p}_z$ in q. Therefore, we can claim that

$$Sim_{RCS}(1^n, index^{p_z}, RCS^I_f, f) \overset{c}{\equiv} \{view^f_{RCS} \\ (index^{p_z}, [(\acute{p_z})_i], n), output^f(index^{p_z}, [(\acute{p_z})_i]). \quad (4)$$

□

Theorem 2. *The protocol IBSvI is simulation secure and securely computes the functionality f, when the party KM is corrupted by $\mathcal{A}_{KM}$ in the presence of semi-honest adversaries.*

Proof. The simulation for the case when KM is corrupted is presented in Protocol 7. After receiving 1^n, KM works as follows:

1. $\mathcal{S}_1$ chooses a uniformly random number $\hat{r} \in \{0,\cdots,2\omega-2\}$.
2. $\mathcal{S}_1$ chooses 2ω uniformly random numbers $\hat{y}_j \in \{-v,\cdots,v\}$, where $v = Max_\alpha(2^\ell-1)(2^\phi-1)$, that contains $\hat{r}$ zeros.
3. $\mathcal{S}_1$ encrypts $\hat{y}_j$ and sends the permuted encryptions $[\hat{y}_j]$ to KM.
4. KM calls the KM_{f_1} functionality to obtain encryptions $[I_j]$ and send them to $\mathcal{S}_1$.
5. $\mathcal{S}_1$ performs UnPerm$[I_j]$, and outputs $[Ans_j] \leftarrow$Half$([I_j])$.

The simulation and the real view can be written as:

$$Sim_{KM}(1^n, \phi, KM_f, f) = (\phi; [\hat{y}_j]; [Ans_j]]). \quad (5)$$

The view and output of KM are $view^f_{KM}(index^{p_z}, \phi, n) = (\phi, [y_j])$ and $output^f(index^{p_z}, [(\acute{p_z})_i]) = (\phi, [Ans_j])$. Since $\mathcal{A}_{KM}$ has the decryption key, we need to show that $\mathcal{A}_{KM}$ cannot distinguish between y_j and $\hat{y}_j$. We need to consider following points to prove the security theorem:

1. The values of $(index^{p_z})_i$ and $(p_z)_i$: as it is presented in Protocol 2, both indices $(index^{p_z})_i$ and $(p_z)_i$ are XORed with uniformly distributed random r in $RCS^I_{f_1}$.
2. The bit-lengths of $(index^{p_z})_i$ and $(p_z)_i$: to hide the bit-lengths of $(index^{p_z})_i$ and $(p_z)_i$, a fix bit-length solution is suggested, where $(index^{p_z})_i$ and $(p_z)_i$ are ℓ bits for every entry.
3. Number of attributes α in q: the value of $alpha$ has a direct effect on the upper and lower bounds of y_j. To prevent $\mathcal{A}_{KM}$ to learn about the α, as it is shown in $RCS^I_{f_3}$, RCS multiplies $[y]_j$ by the difference between α and the maximum number of attributes that QI can put in q, Max_α.
4. Number of zeros in y_j: by learning number of zeros from multiple y_j, $\mathcal{A}_{KM}$ might be able to distinguish between real y_j and $\hat{y}_j$. $RCS^I_{f_4}$ randomizes the number of zeros by inserting a random number of zeros to y_j.

Randomizing the stated four properties guarantees that $\mathcal{A}_{KM}$ cannot distinguish between y_j and $\hat{y}_j$, thus:

$$Sim_{KM}(1^n, \phi, KM_f, f) = \{view^f_{KM} \\ (index^{p_z}, \phi, n), output^f(index^{p_z}, [(\acute{p_z})_i])\}. \quad (6)$$

□

4.2 Security of IBSvII

In IBSvII, computation of Ans_j does not require collaboration of RCS and KM; thus, RCS can compute the final binary vector Ans_j without any communication. To prove the security of Protocol 3, we need to show that RCS cannot learn anything from data.

Theorem 3. *The protocol IBSvII is simulation secure and securely computes the functionality f, when the party RCS is corrupted by adversary $\mathcal{A}_{RCS}$ in the presence of semi-honest adversaries.*

Table 3: Computational complexity of the searching protocols.

Protocols	Addition	Exponentiation	Multiplication	Encryption	Decryption
IBSvI	$(3\alpha\omega\ell+\alpha\omega)/\acute{\rho}$	$(\alpha\omega)_{\log(\theta r_z)}/\acute{\rho}$	0	3ω	$2\omega/\acute{\rho}$
	$O(\alpha\omega\ell/\acute{\rho})$		—	$O(\omega)$	$O(\omega/\acute{\rho})$
IBSvII	$(\alpha\omega\ell)/\hat{\rho}$	0	$(2\alpha\omega\ell+\alpha\omega)/\hat{\rho}$	0	0
	$O(\alpha\omega\ell/\hat{\rho})$		$O(\alpha\omega\ell/\hat{\rho})$	—	—

Proof. We need to show that RCS is unable to computationally distinguish between the truly generated messages given from $\mathcal{S}_3$, the simulator of QI, and randomly generated data. Given $index^{p_z}$ and 1^n (security parameters), RCS in IBSvII works as follows:

1. $\mathcal{S}_3$ randomly generates u $\hat{p}_z \in \mathbb{N}$, where $u \leq \alpha$, u random meta data, and a random $\hat{Q_T}$. Then, $\mathcal{S}_2$ forms $[q] \leftarrow \{\hat{Q}_T, [(\hat{p}_z)_i]||\hat{MD}_i\}$, where $i \in \{0, \cdots, u \leq \alpha - 1\}$, ans send $[q]$ to RCS.
2. RCS executes $RCS^I_{f_i}, i \in \{1, \cdots, 3\}$, and then outputs $[Ans_j]$.

The output of the simulation can be presented as: $Sim_{RCS}(1^n, index^{p_z}, RCS^{II}_f, f) = (index^{p_z}, [(\hat{p}_z)_i]; [\hat{Ans}_j], \phi)$. The real view of RCS is $view^f_{RCS}(index^{p_z}, [(\acute{p}_z)_i]) = (index^{p_z}, [(\acute{p}_z)_i])$ and the real view of the output is $output^f(index^{p_z}, [(\acute{p}_z)_i]) = ([Ans_j], \phi)$. Clearly, the encryptions $[(\hat{p}_z)_i]$ and $[(\acute{p}_z)_i])$, and $[Ans_j]$ and $[\hat{Ans}_j]$ are indistinguishable because of using semantically secure crypto-scheme. Meta-data in q also does not reveal any information about the attributes. Therefor, we can claim that

$$Sim_{RCS}(1^n, index^{p_z}, RCS^I_f, f) \overset{c}{\equiv} \{view^f_{RCS} (index^{p_z}, [(\acute{p}_z)_i]), output^f(index^{p_z}, [(\acute{p}_z)_i]). \tag{7}$$

□

5 PERFORMANCE ANALYSES

5.1 Complexity Analysis

Table 3 shows the computational complexity of our protocols for searching with multiple attributes (IBSvI and IBSvII). In Table 3, we present x number of exponentiations with y-bit exponents as $(x)_y$. As it is illustrated in Table 3, the complexity of IBSvI in terms of the number of additions is linear to the number of attributes in q, bit-length of attributes, and the number of records in the database. Moreover, Table 3 shows how applying data packing reduces the number of homomorphic additions and decryptions. Table 3 also shows IBSvII does not require any encryption and decryption; however, it requires performing $(2\alpha\omega\ell+\alpha\omega)/\hat{\rho}$ homomorphic multiplications, where $\hat{\rho}$ is the number of messages that can be packed by deploying batching.

Table 4 summarizes the communicational complexity of the searching protocols in terms of data transmission, communication round, and storage complexity. According to Table 4, communication round for IBSvI is constant, and it is independent of the bit-length of inputs and the number of attributes in q. Moreover, Table 4 shows that RCS in IBSvII computes Ans_j without any communication with KM. Furthermore, data transmission needed is independent of the number of AND conjunctions used in the query.

Table 4 shows the data transmission needed for the secure searching protocols per party, RCS, and KM, where the complexity of IBSvI is $O(\omega)$ and independent of the number of attributes. Recall that there is no communication between RCS and KM in IBSvII, and clearly, no data is transmitted between these two parties. The batching technique significantly reduces computational costs. Table 4 shows the size of each protocol's index table. The value of $\acute{\rho}$ depends of bit-length of ℓ, where smaller ℓ results in larger $\acute{\rho}$.

Table 4: Communicational and space complexity of the searching protocols. Data transmission complexity is denoted as Data.Trans, communication round as Com.Rond, and index size as IS.

Protocols	Data.Trans	Com.Rond	IS
IBSvI	$4\omega/\acute{\rho}$	1	$\omega\ell\alpha/\acute{\rho}$
IBSvII	0	0	$\omega\ell\alpha$

5.2 Experimental Results

In this section, we present the experimental results of implementing IBSvI and IBSvII. First, we show the run-times of the introduced searching protocols for different α and φ values. Then, we compare

Table 6: Run-times and communication costs of the IBSvI and IBSvII. For (Gentry et al., 2015), we use NA to show that the work does not report the corresponding value.

Protocols	Run-time (second)		Data transmission	index size	communication
	RCS	KM	(megabyte)	(megabyte)	(round)
IBSvI$_{\text{Paillier}}$	0.525	5.06	2.19	8.2	1
IBSvI$_{\text{FV}}$	0.44	0.45	49.45	92.8	1
IBSvII$_{\text{FV}}$	5.54	0	0	92.8	0
(Krell et al., 2017)	4.0		26	75K	NA

their run-times in RCS and KM for different α values. To obtain the run-times of the protocols, we use C++ and external libraries: MPIR, Boost, the Secure Computation Library (SeComLib), and SEAL on a single Linux machine running Ubuntu 14.04 LTS, with a 64-bit microprocessor and 8 GB of RAM. We applied a simple parallelization technique in our implementation (4 threads). The cryptographic key length of the Paillier is chosen according to NIST standards (Barker et al., 2007), which are valid until 2030. Table 5 lists the parameters and their values in our implementation.

Table 6 compares the run-times of the IBSvI, IBSvII, and existing works in RCS and KM. To show the trade-off between computational and communicational costs when using different encryption schemes, we also provide the run-time of IBSvI when FV is used, denoted as IBSvI$_{\text{FV}}$. Note that Paillier cannot be used in IBSvII because it does not support the ciphertext multiplications. Table 6 points out both IBSvI$_{\text{Paillier}}$ and IBSvI$_{\text{FV}}$ demand significantly low computational resources from RCS. These results show that Paillier equipped with data packing and FV with enabled batching have the same performance in performing homomorphic addition. However, there is a noticeable difference between the run-times of IBSvI$_{\text{Paillier}}$ and IBSvI$_{\text{FV}}$ in RCS, which means the decryption function in FV is more efficient than Paillier because of being able to pack more messages into a single ciphertext, $\hat{\rho} > \acute{\rho}$. Table 6 presents the run-time of IBSvII that is roughly similar to the total run-time of IBSvI$_{\text{Paillier}}$. Considering these results, we can conclude that, contrary to popular belief, using fully homomorphic crypto-schemes is more efficient than partially homomorphic ones in certain cases.

Table 6 shows the run-time of secure searching in (Krell et al., 2017), where only the cost of finding the matching records in that work is considered. According to Table 6, the work in (Krell et al., 2017) is also efficient in finding the desired records at the costs of leaking access pattern with an index table size of 75 Gigabytes. Note that the network cost of searching is not provided separately in (Krell et al., 2017); therefore, we put the total network cost (searching and retrieving) in Table 6.

Table 5: Parameters and their values.

Parameter	Symbol	Value
Bit-size of inputs	ℓ	15 bits
Number of records in RCS	ω	10^5
Security parameter	κ	112 bits
Number of queries to obtain FPR		10^7
Capacity of a package (IBSvI)	$\acute{\rho}$	89
Capacity of a package (IBSvII)	$\hat{\rho}$	2048

6 CONCLUSION

Searching in encrypted databases is a challenging task due to the complexity introduced by encryption. In this work, we focus on a medical setting where institutions would like to use the data collected by smart wearables from several vendors to analyze them for the well-being of the patients. In order to make the system usable in practice, we propose a two-step procedure: 1) creation of index tables at the time of uploading data from vendors to the cloud storage unit, and 2) executing queries using these tables. Our complexity analysis and experimental results on a large dataset clearly show the contribution of our work: the performance of the system is outstanding. It is also worth mentioning that our idea of creating index tables can be generalized to other application settings, introducing a scalable and efficient search mechanism for encrypted databases where data will later be processed under encryption.

REFERENCES

Akin, I. H. and Sunar, B. (2015). On the difficulty of securing web applications using cryptdb. *IACR Cryptology ePrint Archive*, 2015:82.

Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. (2007). Nist sp800-57: Recommendation for key management part 1: General (revised). *NIST, Tech. Rep.*

Bellare, M., Boldyreva, A., and O'Neill, A. (2007). Deterministic and efficiently searchable encryption. In *Advances in Cryptology - CRYPTO 2007, 27th Annual*

International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings, pages 535–552.

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 321–334.

Boneh, D., Crescenzo, G. D., Ostrovsky, R., and Persiano, G. (2004). Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 506–522.

Boneh, D., Kushilevitz, E., Ostrovsky, R., and III, W. E. S. (2007). Public key encryption that allows PIR queries. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 50–67.

Bösch, C., Tang, Q., Hartel, P. H., and Jonker, W. (2012). Selective document retrieval from encrypted database. In *Information Security - 15th International Conference, ISC 2012, Passau, Germany, September 19-21, 2012. Proceedings*, pages 224–241.

Byun, J. W., Rhee, H. S., Park, H., and Lee, D. H. (2006). Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Secure Data Management, Third VLDB Workshop, SDM 2006, Seoul, Korea, September 10-11, 2006, Proceedings*, pages 75–83.

Chang, Y. and Mitzenmacher, M. (2005). Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 442–455.

Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M. (1995). Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 41–50.

Chung, K., Kalai, Y. T., and Vadhan, S. P. (2010). Improved delegation of computation using fully homomorphic encryption. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 483–501.

Curtmola, R., Garay, J. A., Kamara, S., and Ostrovsky, R. (2011). Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934.

Ding, X., Liu, P., and Jin, H. (2017). Privacy-preserving multi-keyword top-k similarity search over encrypted data. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1.

Erkin, Z., Veugen, T., Toft, T., and Lagendijk, R. L. (2012). Generating private recommendations efficiently using homomorphic encryption and data packing. *IEEE Trans. Information Forensics and Security*, 7(3):1053–1066.

Fan, J. and Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178.

Gentry, C., Halevi, S., Jutla, C. S., and Raykova, M. (2015). Private database access with he-over-oram architecture. In *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, pages 172–191.

Goh, E. (2003). Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216.

Goldreich, O. (2004). *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press.

Goldreich, O. and Ostrovsky, R. (1996). Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473.

Goldschmidt, P. G. (2005). HIT and MIS: implications of health information technology and medical information systems. *Commun. ACM*, 48(10):68–74.

Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98.

Guo, Z., Zhang, H., Sun, C., Wen, Q., and Li, W. (2017). Secure multi-keyword ranked search over encrypted cloud data for multiple data owners. *Journal of Systems and Software*.

Han, F., Qin, J., Zhao, H., and Hu, J. (2014). A general transformation from KP-ABE to searchable encryption. *Future Generation Comp. Syst.*, 30:107–115.

Johnson, M. E. (2009). Data hemorrhages in the healthcare sector. In *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, pages 71–89.

Kamara, S., Papamanthou, C., and Roeder, T. (2012). Dynamic searchable symmetric encryption. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 965–976.

Krell, F., Ciocarlie, G., Gehani, A., and Raykova, M. (2017). Low-leakage secure search for boolean expressions. In *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, pages 397–413.

Li, M., Li, J., and Huang, C. (2012). A credible cloud storage platform based on homomorphic encryption. *Netinfo Security*, 12(9):35G40.

Meingast, M., Roosta, T., and Sastry, S. (2006). Security and privacy issues with health care information technology. In *Engineering in Medicine and Biology Soci-*

ety, 2006. EMBS'06. 28th Annual International Conference of the IEEE, pages 5453–5458. IEEE.

Miyoshi, R., Yamamoto, H., Fujiwara, H., and Miyazaki, T. (2017). Practical and secure searchable symmetric encryption with a small index. In *Secure IT Systems - 22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8-10, 2017, Proceedings*, pages 53–69.

Ostrovsky, R. (1990). Efficient computation on oblivious rams. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 514–523.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 223–238.

Popa, R. A., Redfield, C. M. S., Zeldovich, N., and Balakrishnan, H. (2012). Cryptdb: processing queries on an encrypted database. *Commun. ACM*, 55(9):103–111.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 457–473.

Song, D. X., Wagner, D., and Perrig, A. (2000). Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*, pages 44–55.

Stefanov, E., Papamanthou, C., and Shi, E. (2014). Practical dynamic searchable encryption with small leakage. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*.

Treskes, R. W., van Winden, L. A., van Keulen, N., Atsma, D. E., van der Velde, E. T., van den Akker-van Marle, E., Mertens, B., and Schalij, M. J. (2017). Using smart technology to improve outcomes in myocardial infarction patients: Rationale and design of a protocol for a randomized controlled trial, the box. *JMIR research protocols*, 6(9).

Troncoso-Pastoriza, J. R., Katzenbeisser, S., Celik, M. U., and Lemma, A. N. (2007). A secure multidimensional point inclusion protocol. In *Proceedings of the 9th workshop on Multimedia & Security, MM&Sec 2007, Dallas, Texas, USA, September 20-21, 2007*, pages 109–120.

Xiong, A.-P., Gan, Q.-X., He, X.-X., and Zhao, Q. (2013). A searchable encryption of cp-abe scheme in cloud storage. In *Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2013 10th International Computer Conference on*, pages 345–349. IEEE.

Zhang, W., Lin, Y., Xiao, S., Wu, J., and Zhou, S. (2016). Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing. *IEEE Trans. Computers*, 65(5):1566–1577.