

Privacy-Preserving Data Analytics as an Outsourced Service

Florian Kerschbaum
SAP Research
Karlsruhe, Germany

florian.kerschbaum@sap.com

Julien Vayssière
SAP Research
Brisbane, Australia

julien.vayssiere@sap.com

ABSTRACT

Two sets of privacy requirements need to be fulfilled when a company's accounting data is audited by an external party: the company needs to safeguard its data, while the auditors do not want to reveal their investigation methods. This problem is usually addressed by physically isolating data and auditors during the course of an audit. This approach however no longer works when auditing is performed remotely.

We present a searchable encryption scheme for outsourcing data analytics. In our scheme the data owner needs to encrypt his data only once and ship it in encrypted form to the data analyst. The data analyst can then perform a series of queries for which he must ask the data owner for help in translating the constants in the queries.

Our searchable encryption schemes allows keyword searches and range queries. Furthermore it allows queries to reuse the results of previous queries as tokens and thereby make dependent queries without interaction. Nevertheless our scheme is provably secure.

Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection—*Cryptographic controls*; E.3 [Data Encryption]: Public key cryptosystems

General Terms

Algorithms, Security

Keywords

Searchable Encryption, Outsourcing, Data Analytics, Remote Auditing

1. INTRODUCTION

We introduce the problem addressed in this paper by first presenting how financial auditing is traditionally performed, then exploring the privacy issues raised by the provision of

auditing services remotely. We then outline the proposed solution, which makes remote auditing possible without sacrificing privacy.

The job of a financial auditor is to analyze a set of accounting documents in order to establish that these documents are a fair representation of the business conducted by the company being audited [1]. In addition, a financial auditor may also be asked to look for indicators of fraud [2].

From an abstract point of view, the auditor queries the accounting data and analyses the results. Even though a large amount of queries may be generated in the course of an audit, the final outcome of the audit can be as small as a few paragraphs in the company's annual report stating that everything was in order. The final report therefore details *what* was found and not *how* it was found.

Each of the two parties, the auditor and the audited company, have privacy requirements towards the other party. The company wants to preserve the privacy of the data it lets the auditor access for the purpose of the audit. On the other hand, what makes the auditor effective and efficient in his work are the queries he runs, which are his know-how and intellectual property. The challenge is therefore to ensure both the privacy of the data and the privacy of the queries.

During the course of the audit, the external auditors perform their work strictly on the premises of the company being audited, where they are given their own private office space. Auditors typically do not access the IT systems of the company directly, but are rather given a copy of any data they need to run queries on. The copy of the data is destroyed once the audit is over.

This model satisfies the two privacy constraints outlined previously. In both cases, the combination of IT and physical security, most importantly physical isolation of the auditors, prevent data from leaving the office where the audit takes place and protects the privacy of the work methods of the auditors.

Because it is based on moving people, the traditional model presented above generates a lot of costs as well as inefficiencies. This is why such audits are typically only performed once a year and their scope is kept minimal.

For companies that use enterprise information systems, opportunities exist however for lowering costs and improving efficiency of audits. Since all data, including in most cases scans of original paper documents, exist in digital form, why not let the external auditors perform the audit remotely? In other words, why not move either data and queries instead of moving people?

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SWS'08, October 31, 2008, Fairfax, Virginia, USA.

Copyright 2008 ACM 978-1-60558-292-4/08/10 ...\$5.00.

Two models are possible here. Either the data is moved from the company to the auditor who then performs local queries on the data, or the queries are sent remotely to the company which returns results to the auditors. In both cases, privacy is weakened when compared to the traditional model.

In the first case, the company loses control over its data once it is sent to the auditors. Legal measures and data governance frameworks may provide a promise of privacy, but not the assurance [22] of privacy. The same reasoning applies to remote queries sent by the auditor and executed on the company’s IT systems: auditors lose control over the privacy of their investigation methods.

1.1 Overview of our solution

The problem addressed in this paper is therefore that of performing remote auditing, a specialized case of data analytics, without sacrificing the privacy of either the data or the queries.

We describe an encryption scheme that can be used to encrypt the data, such that nothing, but the result from the queries will be revealed. In our scheme, the data owner needs to encrypt his data only once and ship it in encrypted form to the data analyst. The analyst can then perform a series of queries for which he must ask the data owner for help in translating the constants in the queries.

Although our encryption schemes reveals the access pattern, i.e. reveals which documents match the query, the query remains entirely private, since it is performed on the data analyst’s side. The results of the data analysis, such as an identified fraud case are then shipped back to the data owner.

We introduce our query language over data organized in tables in section 2. After presenting definitions in section 3, we detail our proposed searchable encryption scheme for outsourcing data analytics (SEODA) in section 4. A number of extensions are considered in section 5. The security of our scheme is examined in section 6. Related work is presented in section 7, and conclusions in section 8. Implementation details and performance figures are given in appendix A.

2. QUERY LANGUAGE AND SETUP

This section presents our query language using fraud auditing examples. We also address query result reusability and compare our approach to previous work.

2.1 Query Language

The auditors would be able to use the following language for querying the data:

Let $t = \langle d^1, d^2, \dots, d^n \rangle$ be a n-tuple (row) in the ledger data. Denote $t.d^i$ the i-th data item of row t . For simplicity we consider a flattened, non-normalized form of the data, where all data tables have been condensed into one flat table. Let c^i be any constant query string for $t.d^i$. The grammar G for a query expression $expr$ is as follows:

```

expr := expr  $\wedge$  expr | expr  $\vee$  expr | sexpr
sexpr := t.di op ci | t.di ops t'.di
op := ops | < | >
ops := ==

```

This grammar implies range queries ($c^i < t.d^i \wedge t.d^i < c^{i'}$) and keyword searches ($c^i == t.d^i$). We write

$expr \models t$, if $expr$ matches the record t . The footprint ($\mathbb{RC}, \mathbb{IC}, \mathbb{FK}$) of an expression $expr$ is its set \mathbb{RC} of used constants in range queries, its set \mathbb{IC} of used constants in identity queries and its set \mathbb{FK} of used foreign fields or keys. E.g. the query expression $t.d^1 > 2 \wedge t.d^1 < 6 \wedge t.d^2 == 4 \wedge t.d^3 == t'.d^3$ has the footprint $(\{2, 6\}, \{4\}, \{t'.d^3\})$.

2.2 Example Queries for Fraud Auditing

Let’s examine on a few examples how such a query language could be used for expressing some of the most common queries in fraud auditing.

A very common fraud pattern is employees misappropriating company funds by creating fake invoices and getting these invoices paid to themselves. In order to lower the risk of being caught, fraudsters commonly use two techniques. First, the amount of the fake invoice is kept just under the threshold above which an explicit approval from a manager would be required. In addition, fraudsters often create invoices for types of supplies that do not require a Goods Receipt (GR) document [24]. This document is produced when physical goods are delivered to the company, and is required as a proof of delivery for triggering the payment of the invoice. However, for immaterial supplies such as services, no GR is produced, which weakens the invoice verification process. Assuming the threshold is \$10000 and $amount$ is the field representing an invoice amount in a table row t of invoices paid by the company, the query could be:

```

t.amount > 9500  $\wedge$  t.amount < 10000  $\wedge$  t.GRRequired
== false

```

Which informally can be expressed as “give me all invoices with an amount \$500 below the threshold and for which no Goods Receipt document is required”.

A variation on the fraud pattern above is for a fraudster to create a fake storefront supplier and issue invoices for payment into an arbitrary bank account. Cases are documented where the fraudster used the same bank account as the one the company pays his salary into [23]. If we assume a field $invoiceBankAccount$ in a table row t of invoices and a field $employeeBankAccount$ in a table row t' of employees, the query for the suspicious employee would be:

```

t.employeeBankAccount == t'.invoiceBankAccount

```

2.3 Query Result Reusability

The language implies an important feature of the encryption scheme: comparing for equality the data items of two tuples. If a query has returned a set of tuples, each data item in this tuple can be used in subsequent keyword searches without loss of privacy. We call this feature query result reusability.

Full query result reusability, i.e. the ability to reuse the query results in arbitrary range queries, contradicts provable security. The querier could always sort two resulting ciphertexts for tuples t, t' by using a returned query token $t'.d^i$ on the other ciphertext for the query $t.d^i < t'.d^i$. As a designer, one can then either change the security definition and restrict the attacker to not obtain query tokens that decrypt his challenge ciphertexts which would be very limiting or restrict the search power of the returned query tokens.

Another differentiating feature of the encryption system is that queries are not revealed to the encrypting party.

Nevertheless the querying party can search for any range it chooses. In a completely non-interactive system these are conflicting goals. If the querier can (non-interactively) search for any range he intends to, he can binary search for the encrypted value and thereby break any encryption secure against a polynomially-bound adversary.

We therefore chose to make the translation of constants into query tokens an interactive protocol, but one that does not reveal its inputs. The query token protocol for the querier is a privacy-preserving protocol that protects the constant from the encrypting party (and the secret key from the querier). The encryption scheme preserves the privacy of the query.

2.4 Improvements over Previous Work

Previous results consider range queries for searchable encryption, but none considered query result reusability and query privacy. The work of Shi et al. [20] presents an encryption scheme that is very efficient for range queries. Its ciphertext size is logarithmic in the size of the domain \mathbb{D} of the encrypted values. The Boneh-Waters [6] encryption scheme supports queries for arbitrary subsets and opposed to Shi et al. does not forcibly reveal the resulting data item on a matching query (match-concealing vs. match-revealing). It is therefore better suited for query privacy, but still a query token may reveal the data item queried for. Its ciphertext size has the same factor as the ciphertext size of our scheme: $O(\mathbb{D})$.

Another competing approach is to use private information retrieval (PIR) [7, 9, 18] over single-key encrypted data. One can even realize range queries and query privacy with similar techniques and complexity to our encryption scheme. We improve over those techniques by reducing the communication complexity by a factor linear in the number of queries. Furthermore we know from [21] that the limiting factor in PIR is computational complexity and in the PIR approach the data owner needs to carry the computational load while in our approach the data analyst carries the higher computational load. Boneh et al. [5] extend PIR to search on public-key encrypted data, but at a further performance expense and without the possibility for range queries.

Let l be the number of tuples in the database. Our encryption scheme has key size $O(|t|)$ (where $|t|$ is the number of fields in a tuple), ciphertext size $O(\mathbb{D}|t|^2l)$, query token size $O(1)$, encryption complexity $O(\mathbb{D}|t|^2l)$, range query complexity $O(\mathbb{D}|t|l)$ and keyword search complexity $O(|t|l)$.

3. DEFINITIONS

This section introduces the definitions used later in the description of our encryption scheme and also gives an explicit definition of the security of our solution.

3.1 Encryption Scheme

DEFINITION 1. *A searchable encryption scheme for outsourcing data analytics (SEODA) consists of the following polynomial-time algorithms or protocols:*

1. **Setup**(k, Γ): *Takes a security parameter k and tuple definition Γ and outputs a secret SK and a public security parameter PP .*
2. **Encrypt**(SK, t): *Takes a secret key SK and a tuple t (adhering to Γ) and outputs a ciphertext C .*

3. **PrepareRangeQuery**[(rc^i, PP), (SK)]: *Is a protocol between the data analyst DA and the data owner DO . The analyst inputs a constant rc^i and a public security parameter PP and the owner inputs a secret key SK . The output at the analyst is a range query token RQ and the data owner receives no output. The protocol hides the inputs, such that the analyst will learn nothing about SK and the data owner nothing about c .*
4. **PrepareIdentityQuery**[(ic^i, PP), (SK)]: *Is a protocol between the data analyst DA and the data owner DO . The analyst inputs a constant ic^i and a public security parameter PP and the owner inputs a secret key SK . The output at the analyst is an identity query token IQ and the data owner receives no output. The protocol hides the inputs, such that the analyst will learn nothing about SK and the data owner nothing about c .*
5. **Analyze**($PP, C, \mathbb{RQ}, \mathbb{IQ}, expr$): *Takes a public security parameter PP , a ciphertext C , a set of range query tokens \mathbb{RQ} , a set of identity query tokens \mathbb{IQ} and a query expression $expr$ (adhering to the grammar G) and outputs a set \mathbb{IQ}' of identity query tokens.*

For the encryption scheme to be searchable we impose the following consistency constraint. For each tuple $t = \langle c^1, c^2, \dots, c^n \rangle$ (defined by Γ) and each query expression $expr$ with footprint ($\mathbb{RC}, \mathbb{IC}, \mathbb{FK}$), the above scheme must satisfy the consistency constraint from Figure 1.

3.2 Security

DEFINITION 2. *We say that a SEODA scheme \mathcal{E} is secure if all polynomial-time adversaries \mathcal{A} have at most a negligible advantage in the security game defined below.*

- **Setup:** The challenger runs $\text{Setup}(k, \Gamma)$ and passes the public parameter PP to the adversary.
- **Query Phase 1:** The adversary adaptively outputs a sequence of either
 - a plain text tuple t_1, t_2, \dots, t_{q_1} ,
 - a range query constant $rc_1^i, rc_2^i, \dots, rc_{q_2}^i$, or
 - an identity query constant $ic_1^i, ic_2^i, \dots, ic_{q_3}^i$.

The challenger responds corresponding to the type of the query with either

- the ciphertext $C = \text{Encrypt}(SK, t)$.
- the range query token $RQ = \text{PrepareRangeQuery}[(rc^i, PP), (SK)]$.
- the identity query token $IQ = \text{PrepareIdentityQuery}[(ic^i, PP), (SK)]$.
- **Challenge:** The adversary outputs two plain-text tuples t_0^* and t_1^* subject to the following restrictions
 - none of t_j 's constants $t_j.c^i$ matches any constant of the challenge plain texts $t_{\{0,1\}}^*$, i.e.

$$\forall j \in 1, 2, \dots, q_1 \forall i. t.c^i \neq t_0^*.c^i \wedge t.c^i \neq t_1^*.c^i$$

$$\text{Analyze}(PP, C, \mathbb{RQ}, \mathbb{IQ}, \text{expr}) = \begin{cases} \{\text{PrepareIdentityQuery}[(c, PP), (SK)] \mid \forall c \in \{c^1, c^2, \dots, c^n\}\} & \text{if } \text{expr} \models t \\ \perp & \text{w.h.p., otherwise} \end{cases}$$

where

$$PP, SK = \text{Setup}(k, \Gamma)$$

$$C = \text{Encrypt}(SK, t)$$

$$\mathbb{RQ} \supset \{\text{PrepareRangeQuery}[(c, PP), (SK)] \mid c \in \mathbb{RC}\}$$

$$\mathbb{IQ} \supset \{\text{PrepareIdentityQuery}[(c, PP), (SK)] \mid c \in \mathbb{IC} \cup \mathbb{FK}\}$$

Figure 1: Consistency Constraint

- no range query constant rc_j^i can distinguish the challenge plain texts $t_{\{0,1\}}^*$, i.e.

$$\forall j \in 1, 2, \dots, q_2. rc_j^i < \min(t_0^*.c^i, t_1^*.c^i) \vee \\ rc_j^i > \max(t_0^*.c^i, t_1^*.c^i)$$

- no identity query constant ic^i matches the challenge plain texts $t_{\{0,1\}}^*$, i.e.

$$\forall j \in 1, 2, \dots, q_3. ic_j^i \neq t_0^*.c_i \wedge ic_j^i \neq t_1^*.c_i$$

The challenger flips a coin $b \in \{0, 1\}$ and encrypts t_b^* under SK . The ciphertext $C^* = \text{Encrypt}(SK, t_b^*)$ is passed to the adversary.

- **Query Phase 2:** The adversary continues to adaptively output plain texts, range query constants and identity constants subject to the restrictions above. The challenger responds with the corresponding ciphertexts, range and identity query tokens.

- **Guess:** The adversary outputs a guess b' of b .

An adversary \mathcal{A} 's advantage in the above game is defined as

$$\text{Adv}_{\mathcal{A}} = |\text{Pr}[b = b'] - \frac{1}{2}|$$

4. OUR SEODA SCHEME

For simplicity of the exposition we construct a SEODA scheme for 1-dimensional tuples $t = \langle c \rangle$ in this section. Let $\mathbb{D} = [d_1, d_2]$ be the domain of c and $\tau = |\mathbb{D}| = d_2 - d_1 + 1$ be the size of the domain. The next section will extend this to multi-dimensional tuple definitions.

Setup(k, Γ): The data owner DO chooses primes p, q , and $o, p = 2q + 1 = 4o + 3$, such that $o > \tau$, that the DDH problem is assumed to be infeasible in G_q^* (see 6.2) and that the DL problem is assumed to be infeasible in G_p^* (see 6.1). He randomly selects two secret keys $x_{<}$ and $x_{=}$ from \mathbb{Z}_q^* . He further randomly selects a group element g from \mathbb{Z}_p^* of order $2q$ (a generator of \mathbb{Z}_p^*). The public parameter PP is p and g and is given to the data analyst.

Encrypt(SK, t): The data owner creates a ciphertext

$$\beta = c^{2x_{=}} \bmod q$$

For each $d \in \mathbb{D}$ he randomly selects an element r in \mathbb{Z}_q^* and an element r' in \mathbb{Z}_p^* and computes

$$\gamma = d^{2x_{<}} \bmod q$$

$$e = g^{2r} \bmod p$$

$$e' = \begin{cases} \beta^2 g^{2r\gamma} \bmod p & \text{if } c < d \\ r' & \text{otherwise} \end{cases}$$

Let (e_j, e'_j) be the encryption (e, e') for $d_j \in \mathbb{D}$ as described above. Then, let $\widehat{C}_{<} = (e_1, e'_1), (e_2, e'_2), \dots, (e_\tau, e'_\tau)$ be the encryption for the entire “smaller-than” range query. The data owner chooses a random permutation $\Pi_{<}$ of $[1, \tau]$. Let $C_{<}$ be an element-wise permutation of the encryptions of $\widehat{C}_{<}$ according to $\Pi_{<}$.

Let Π be a fixed random permutation of \mathbb{Z}_q^* known to the data owner and analyst and $\Pi(a)$ be the element at position a in the permutation. In practice, one can use e.g. a cryptographic hash function of equal bit length. The function does not have to be one-way or trap-door. Again, for each $d \in \mathbb{D}$ he randomly selects an element r in \mathbb{Z}_q^* and an element r' in \mathbb{Z}_p^* and computes

$$\gamma = d^{2x_{<}} \bmod q$$

$$e = g^{2r} \bmod p$$

$$e' = \begin{cases} \Pi(\beta)^2 g^{2r\gamma} \bmod p & \text{if } c > d \\ r' & \text{otherwise} \end{cases}$$

Let (e_j, e'_j) be the encryption (e, e') for $d_j \in \mathbb{D}$ as described above. Then, let $\widehat{C}_{>} = (e_1, e'_1), (e_2, e'_2), \dots, (e_\tau, e'_\tau)$ be the encryption for the entire “greater-than” range query. The data owner chooses another random permutation $\Pi_{>}$ of $[1, \tau]$. Let $C_{>}$ be an element-wise permutation of the encryptions of $\widehat{C}_{>}$ according to this permutation $\Pi_{>}$ of the “greater-than” range queries.

For identity queries the data owner randomly selects two elements r and r' in \mathbb{Z}_q^* and computes

$$e = g^{2r} \bmod p$$

$$e' = \beta^2 g^{2r\beta} \bmod p$$

Let $C_{=} = (e, e')$ be the encryption for the identity query.

The ciphertext C for the entire tuple t is $C = C_{<}, C_{>}, C_{=}$.

PrepareRangeQuery($(rc^i, PP), (SK)$): The PrepareRangeQuery operation is an interactive, privacy-preserving protocol between the data owner DO and the data analyst DA . The data analyst has as input a constant rc (we can drop the superscript i , since we assume one-dimensional tuples) he wishes to use in a range query as upper or lower bound.

He chooses a random element r in \mathbb{Z}_{q-1}^* such that r has a multiplicative inverse r^{-1} in \mathbb{Z}_{q-1}^* , i.e. $rr^{-1} \bmod (q-1) = 1$, and he sends the following to the data owner:

$$DA \longrightarrow DO : rc^{2r} \bmod q$$

The data owner has as input the secret key SK and in particular $x_{<>}$. He returns to the data analyst:

$$DO \longrightarrow DA : (rc^{2r})^{x_{<>}} = rc^{2rx_{<>}} \bmod q$$

The data analyst computes

$$DA : RQ = (rc^{2rx_{<>}})^{r^{-1}} = rc^{2x_{<>}} \bmod q$$

Note that the range query token RQ corresponds to the intermediate key γ in the Encrypt operation. The data analyst outputs RQ ; the data owner has no output. During the protocol, loosely speaking, the data analyst has learnt nothing about $x_{<>}$ and the data owner has learnt nothing about rc .

PrepareIdentityQuery[(ic^i , PP), (SK)]: The PrepareIdentityQuery operation is the duplicate of PrepareRangeQuery for identity queries. The data analyst has as input a constant ic he wishes to use in an identity query. The data owner has as input the secret key SK , in particular $x_{=}$. The data analyst has as output an identity query token IQ and the data owner has no output. The data analyst chooses a random element r in \mathbb{Z}_{q-1}^* , such that r has a multiplicative inverse r^{-1} in \mathbb{Z}_{q-1}^* , i.e. $rr^{-1} \bmod (q-1) = 1$ and the protocol proceeds as follows:

$$DA \longrightarrow DO : rc^{2r} \bmod q$$

$$DO \longrightarrow DA : (rc^{2r})^{x_{=}} = rc^{2rx_{=}} \bmod q$$

$$DA : IQ = (rc^{2rx_{=}})^{r^{-1}} = rc^{2x_{=}} \bmod q$$

Analyze(PP , C , \mathbb{RQ} , \mathbb{IQ} , $expr$):

We already assume that the tuple t is one-dimensional and we now limit the exposition to either simple range queries $expr := c > rc \wedge c < rc'$ or identity queries $expr := c == ic$. Complex queries will be explained in a later section. It must be ensured that the footprint of $expr$, i.e. the range query tokens RQ and RQ' for rc and rc' and the identity query token IQ for ic , is in the set \mathbb{RQ} and \mathbb{IQ} , respectively.

Let $y = x^2 \bmod p$ be a quadratic residue modulo p . Then $x' = y^{o+1} \bmod p$ is a square root of y and the other square root is $x'' = p - x'$. Let $\sqrt[*]{y} = \min(x', x'')$ be the smaller square root. Note that $\sqrt[*]{y} \leq q$.

Recall that $RQ = rc^{2x_{<>}}$. A range query $expr$ proceeds as follows: Let $C_{<} = (e_1, e'_1), (e_2, e'_2), \dots, (e_\tau, e'_\tau)$ be the “smaller-than” range query part of C . For each (e_i, e'_i) (with $i = 1, 2, \dots, \tau$) the data analyst computes

$$\zeta_i = \sqrt[*]{\frac{e'_i}{e_{iRQ}}} \bmod p$$

Let $C_{>} = (e_1, e'_1), (e_2, e'_2), \dots, (e_\tau, e'_\tau)$ be the “greater-than” range query part of C . For each (e_i, e'_i) (with $i = 1, 2, \dots, \tau$) the data analyst computes

$$\eta_i = \sqrt[*]{\frac{e'_i}{e_{iRQ'}}} \bmod p$$

Recall that Π is a fixed random permutation of \mathbb{Z}_q^* . If $\Pi(\zeta_i) = \eta_j$ for any $i, j \in [1, \tau]$, then the data analyst concludes that $expr \models t$

An identity query $expr := c == ic$ proceeds as follows: Recall that $IQ = ic^{2x_{=}}$. Let $C_{=} = (e, e')$ be the encryption for the identity query. The data analyst computes

$$\theta = \sqrt[*]{\frac{e'}{e^{IQ}}} \bmod p$$

If $\theta = IQ$, then the data analyst concludes that $expr \models t$.

5. EXTENSIONS

5.1 Multiple Dimensions

Our SEODA scheme supports multi-dimensional database tuples of the form $t = \langle c^1, c^2, \dots, c^n \rangle$. In the Setup(k , Γ) operation the data owner chooses n pairs of secret keys $x_{<>}^1, x_{<>}^2, x_{<>}^3, \dots, x_{<>}^n, x_{=}^n$: one key pair $x_{<>}^i, x_{=}^i$ for each data item c^i . In the Encrypt(SK , t) operation the data owner encrypts each c^i with each secret key $x_{=}^i$, i.e. each data item is encrypted with each key. This creates n^2

$$\beta_{i,j} = (c^i)^{2x_{=}^j} \bmod q$$

He then creates intermediate ciphertexts $C_{<}$, $C_{>}$, and $C_{=}$ for each $\beta_{i,j}$ (under each corresponding domain element $d^i \in \mathbb{D}^i$). The resulting ciphertext has size $O(\mathbb{D}n^2)$ and the secret key has size $O(n)$.

The data analyst in the Analyze(PP , C , \mathbb{RQ} , \mathbb{IQ} , $expr$) operation has obtained a query token for a specific element c^i of the tuple. His goal is to obtain an identity query token for element $t.d^i$. He then selects the ciphertext part $C_{<,>=}$ that encrypts $\beta_{i,j}$ (under the domain elements $d^i \in \mathbb{D}^i$) and applies the described algorithm.

5.2 Limiting Linkability

The procedure described in the previous section allows the data analyst to link every element with every other element. He can obtain an identity query token for any column by searching for elements in any other column. Not every database structure requires this, e.g. the following database relations

$$d^1 \mapsto d^2 d^3 \quad d^3 \mapsto d^4 d^5$$

do not require the full search capability. Instead it can be sufficient to be able to search for triples (d^1, d^2, d^3) and (d^3, d^4, d^5) . The data owner can restrict the data analyst to a subset of possible searches by selectively not encrypting all $\beta_{i,j}$. In the example all pairs that are not contained in any triple (e.g. (d^1, d^4) and (d^4, d^1)) can be left out of the ciphertext. Obviously this choice has to be communicated to the data analyst, such that he can make the right selection in the ciphertext.

5.3 Complex Queries

Our SEODA scheme supports the full grammar G not only simple range or identity queries as described in the Analyze(PP , C , \mathbb{RQ} , \mathbb{IQ} , $expr$) algorithm. The data analyst first breaks the query into simple queries of the form described connected by the binary operators \wedge or \vee . He might need to obtain range query tokens for virtual lowest $-\infty$ and upmost elements ∞ to augment range queries of the form $t.d^i < c^i$ or $t.d^i > c^i$ with $t.d^i > -\infty$ or $t.d^i < \infty$, respectively, that cannot be combined otherwise. In a first step he then executes all simple range queries obtaining a result set Ω_j for each simple query expression $expr_j$.

As a second step he combines the sets: For two query expressions $expr$ and $expr'$ connected by a \wedge operation he computes the combined result set $\Omega_\wedge = \Omega \cap \Omega'$. For two query expressions $expr$ and $expr'$ connected by a \vee operation he computes the combined result set $\Omega_\vee = \Omega \cup \Omega'$.

The final result set Ω^* that combines all query expressions is the result of the complex query.

Our complex queries reveal more than the result of a complex query itself, but this is in line with our security definition. Opposite from e.g. [15] the data analyst can always issue parts of the each complex query without additional approval by the data owner and gain the information our complex query reveals, i.e. our complex queries do not reveal more than is accessible to the data analyst anyway.

6. SECURITY

6.1 Discrete Logarithm Problem

For any probabilistic polynomial-time algorithm \mathcal{A} , we define \mathcal{A} 's advantage in solving the Discrete Logarithm problem in \mathbb{Z}_p^* with group generator g of order $2q$ as

$$Adv_{\mathcal{A}}^{DL}(k) = \left| Pr_{a \in \mathbb{Z}_q^*} [\mathcal{A}(p, g, g^{2a}) = 2a] \right|$$

DEFINITION 3. We say that the Discrete Logarithm assumptions (DL) holds for \mathbb{Z}_p^* , if for any probabilistic polynomial-time algorithm \mathcal{A} , $Adv_{\mathcal{A}}^{DL}(k)$ is a negligible function of k .

6.2 Decisional Diffie-Hellman Assumption

For any probabilistic polynomial-time algorithm \mathcal{A} , we define \mathcal{A} 's advantage in solving the Decisional Diffie-Hellman problem in \mathbb{Z}_p^* with group generator g of order $2q$ as

$$Adv_{\mathcal{A}}^{DDH}(k) = \left| Pr_{a, b \in \mathbb{Z}_q^*} [\mathcal{A}(p, g, g^{2a}, g^{2b}, g^{2ab}) = 1] - Pr_{a, b, c \in \mathbb{Z}_q^*} [\mathcal{A}(p, g, g^{2a}, g^{2b}, g^{2c}) = 1] \right|$$

DEFINITION 4. We say that the Decisional Diffie-Hellman Assumption (DDH) holds for \mathbb{Z}_p^* , if for any probabilistic polynomial-time algorithm \mathcal{A} , $Adv_{\mathcal{A}}^{DDH}(k)$ is a negligible function of k .

6.3 Protocol Security

We define security of the protocols for PrepareRangeQuery and PrepareIdentityQuery as semi-honest security for Secure Two-Party Computation protocols [13], i.e. we assume that all parties follow the protocol as described, but try to break its confidentiality (and therefore the confidentiality of the encryption scheme). The view of a party (data owner or data analyst) during a protocol is his input, his coin tosses and the messages he receives. The output is implicit in the view.

DEFINITION 5. The view of a party $X \in \{A, B\}$ during an execution of a protocol Ψ between A and B on inputs (ω_A, ω_B) is denoted

$$VIEW_X^\Psi = \{\omega_X, r, m_1, \dots, m_\phi\}$$

where r represents the outcome of A 's internal coin tosses, and m_i represents the i -th message it has received.

We define the security of a protocol Ψ

DEFINITION 6. Let $f^\Psi(\omega_A, \omega_B) : (\{0, 1\}^*)^2 \mapsto (\{0, 1\}^2)$ be the (ideal) functionality implemented by protocol Ψ . The protocol Ψ is secure in the semi-honest model, if there exists a polynomial-time simulator, denoted S , such that for any probabilistic polynomial-time algorithm \mathcal{A} , $S(\omega_X, f^\Psi(\omega_A, \omega_B))$ is computationally indistinguishable from $VIEW_X^\Psi$:

$$S(\omega_X, f^\Psi(\omega_A, \omega_B)) \stackrel{c}{=} VIEW_X^\Psi$$

We propose the following theorems for the security of the PrepareRangeQuery and PrepareIdentityQuery protocols.

THEOREM 1. In our SEODA scheme, the PrepareRangeQuery protocol is secure in the semi-honest model.

THEOREM 2. In our SEODA scheme, the PrepareIdentityQuery protocol is secure in the semi-honest model.

PROOF. We prove theorem 1 by giving simulators for the views $VIEW_{DO}^{PrepareRangeQuery}$ and $VIEW_{DA}^{PrepareRangeQuery}$ of the data owner DO and the data analyst DA . Recall, that the protocol PrepareRangeQuery implements the ideal functionality $f^{PrepareRangeQuery}((c, p, g), (x_{<>}, p)) = (c^{x_{<>}}, \perp)$ for the data analyst DA and the data owner DO . The simulator for the data owner's view $VIEW_{DO}^{PrepareRangeQuery}$ is the secret SK as input, no random coin tosses and a random element r from the set of quadratic residues \mathbb{QR}_q in \mathbb{Z}_q^* , since rc^2 is a generator of \mathbb{QR}_q . The simulator for the data analyst's view $VIEW_{DA}^{PrepareRangeQuery}$ is the public parameters p and g as input, a random coin toss r , and the message $c^{2rx_{<>}}$. \square

The security of the secret key $x_{<>}$ does not follow from the protocol's security, but rather from the security of the ideal functionality. We propose that the security of the secret key $x_{<>}$ follows from the DL assumption.

PROPOSITION 1. The secret key $x_{<>}$ in the ideal function $f^{PrepareRangeQuery}$ is secure against the data analyst, if the DL problem is infeasible in \mathbb{Z}_p^* .

PROPOSITION 2. The secret key $x_{=}$ in the ideal function $f^{PrepareIdentityQuery}$ is secure against the data analyst, if the DL problem is infeasible in \mathbb{Z}_p^* .

The proof of security for the PrepareIdentityProtocol is identical except that it replaces the secret key $x_{<>}$ with the secret key $x_{=}$.

6.4 Security and Correctness

For simplicity we consider only the two simple queries on 1-dimensional tuples for correctness again.

THEOREM 3. The probability $Pr[\Pi(\zeta_i) = \eta_i | \neg(expr \models t)]$ for a false match in range queries and the probability $Pr[\theta = IQ | \neg(expr \models t)]$ for a false match in identity queries are negligible functions.

Let $expr$ be a range query of the form $c > rc \wedge c < rc'$. We start with the case $expr \models t$. If $expr \models t$, then there is a position i where it holds that $d = rc$ and position j where it holds that $d = rc'$. The data analyst computes

$$\zeta_i = \sqrt{*} \frac{e'_i}{e_i} = \sqrt{*} \frac{(c^{2x_{<>}})^2 g^{2r(d^{2x_{<>}})}}{(g^{2r})(rc^{2x_{<>}})} = c^{2x_{<>}}$$

$$\eta_j = \sqrt{\frac{e'_j}{e_j^{RQ}}} = \sqrt{\frac{\prod(c^{2x=})^2 g^{2r(d^{2x<>})}}{(g^{2r})^{(rc^{2x<>})}} = \prod(c^{2x<>})$$

Then, obviously $\Pi(\zeta_i) = \eta_i$, and $\zeta_i = \text{PrepareIdentityQuery}[(c, PP), (SK)]$. Therefore the correctness condition is fulfilled, if $\text{expr} \models t$. In the other case, if $\neg(\text{expr} \models t)$, then either $e' = r'$ or $d \neq rc$ at all τ positions of $C_<$ and $C_>$, a random element in \mathbb{Z}_p^* and ζ_i will be a random element in \mathbb{Z}_q^* , such that the probability of a false match is

$$\Pr[\Pi(\zeta_i) = \eta_i | \neg(\text{expr} \models t)] = \frac{|\mathbb{D}|^2}{q} = \frac{\tau^2}{2^k}$$

which is a negligible function in k . Therefore $\text{Analyze}(PP, C, \mathbb{RQ}, \mathbb{IQ}, \text{expr}) = \perp$ w.h.p, if $\neg(\text{expr} \models t)$.

Now, let expr be an identity query of the form $c == ic$. If $\text{expr} \models t$, then it holds that $c = ic$ and the data analyst computes

$$\theta = \sqrt{\frac{e'}{e^{IQ}}} = \sqrt{\frac{(c^{2x=})^2 g^{2r(c^{2x=})}}{(g^{2r})^{(ic^{2x=})}} = c^{2x=}$$

Then $\theta = IQ = \text{PrepareIdentityQuery}[(c, PP), (SK)]$, and correctness holds.

If $\neg(\text{expr} \models t)$, then $c \neq ic$ and θ is randomly distributed in \mathbb{Z}_q^* . The probability of a false match

$$\Pr[\theta = IQ | \neg(\text{expr} \models t)] = \frac{1}{q} = \frac{1}{2^k}$$

which is again a negligible function in k . Therefore $\text{Analyze}(PP, C, \mathbb{RQ}, \mathbb{IQ}, \text{expr}) = \perp$ w.h.p, if $\neg(\text{expr} \models t)$.

We state the following theorem about the security of our SEODA scheme.

PROPOSITION 3. *If the Discrete Logarithm assumption holds in \mathbb{Z}_q^* , then the advantage $\text{Adv}_{\mathcal{A}}$ in the security game above for any polynomial-time algorithm \mathcal{A} is a negligible function.*

6.4.1 Intuition

The following illustrates that the security is at most as hard as the Decisional Diffie-Hellman problem. The adversary queries the challenger according to the restrictions in the game. He chooses a random plaintext g^* of order $2o$ in \mathbb{Z}_q^* and obtains the identity query token $IC_g^* = (g^*)^{2x=} \bmod q$. Then he chooses two plain texts (tuples) c_1^* and c_2^* where $\text{wlog } c_1^* = (g^*)^h \bmod q$ and $c_2^* = (g^*)^{h'} \bmod q$ ($h \neq h'$). The challenger responds with one ciphertext c_b^* and the adversary extracts using a range query the identity query token IC_b^* for this ciphertext:

$$\begin{aligned} IC_b^* \in \{ & IC_1^* = (c_1^*)^{2x=} = (g^*)^{2hx=} \bmod q, \\ & IC_2^* = (c_2^*)^{2x=} = (g^*)^{2h'x=} \bmod q \} \end{aligned}$$

Note that $((c_1^*)^{2x=} = (g^*)^{2h}, IC_g^* = (g^*)^{2x=}, IC_1^* = (g^*)^{2hx=})$ is a Diffie-Hellman tuple, i.e. the adversary is exactly given an instance of the DDH problem.

This section illustrates that the security is at most as hard as the Discrete Logarithm problem. The adversary queries the challenger according to the restriction in the game. He obtains a number of identity query tokens $c_1^{2x=}, \dots, c_n^{2x=}$ and finally chooses two plain texts (tuples) c_1^* and c_2^* . The challenger responds with one ciphertext c_b^* and the adversary extracts using a range query the identity query token $IC_b^* = (c_b^*)^{2x=}$ for this ciphertext. If the adversary can obtain $x=$, he can “decrypt” the identity query token IC_b^* .

7. RELATED WORK

Most searchable encryption schemes work in the database as a service (DAS) model [16]. In the DAS model there is a user and a service provider. The service provider offers his database, e.g. over the Internet as a service to the user. The user wants to protect his data against outsiders and the service provider himself. He therefore encrypts the data with a secret key. The problem that arises is that of querying the encrypted data. Neither the data should be revealed nor the query itself. This can be addressed by a searchable encryption scheme.

Examples of such searchable encryption schemes are [8, 10, 11, 12, 25]. All these schemes allow searching for keywords on a secret-key encrypted database without revealing the keyword. Note that for efficiency all schemes leak the *access pattern*, i.e. the documents (or tuples) matching the query. Stronger security requires less efficient solutions, such as oblivious RAM [14].

Keyword searches are important, but to be useful in practice, range queries are indispensable. The problem of range queries has been addressed in [6, 17, 20]. Searchable encryption with range queries is presented in [6, 20]. Both schemes present efficiency improvements for range queries in searchable encryption, but both reveal at least partially the query to the service provider. Therefore a different application than DAS is suggested in [20] where the database owner publishes his data, but only gives decryption keys (for certain ranges) to qualified users. In [17] privacy is protected in a best effort approach based on bucketization. An approach that allows range queries by not permuting the order during encryption is described in [3].

Another strand of work [4, 5] extends searchable encryption to public key encryption. This is useful for an outsourced e-mail service where the user receives documents (or tuples) from other users, but still has the same security requirements as in the DAS model. Keyword searches are described in [4] and private index queries (PIR queries) are described in [5].

Private information retrieval (PIR) [7, 9, 18] allows a querier to ask for an entry in a remote database without revealing the index of this entry. PIR fully hides the access pattern, i.e. the service provider (database) is not aware which document (tuple) was chosen. This can be done with polylogarithmic communication complexity [7], i.e. without transferring the entire database. We already mentioned the drawbacks of this approach: It requires processing each document for each query on the service providers' side which is less practical than transferring the entire database [21].

8. CONCLUSION

We presented a searchable encryption scheme for outsourcing data analytics. This work is strongly motivated by the search for a privacy-assured solution for electronic, outsourced auditing and fraud detection. A data owner wants another party to perform data analytics, e.g. auditing or fraud detection, on his data, but neither the data owner wants to reveal his data nor the analyst his queries. In our scheme the data owner needs to encrypt his data only once and ship it in encrypted form to the data analyst. The data analyst can then perform a series of queries for which he must ask the data owner for help in translating the constants in the queries. Although our encryption schemes re-

veals the access pattern, i.e. reveals which documents match the query, the query remains entirely private, since it is performed at the data analyst's side.

Out of practical considerations we implement both: keyword searches and range queries. The result of a matching query can be reused as a token for keyword search in subsequent queries allowing to make dependent queries. Nevertheless the features of the encryption scheme have been carefully chosen to allow for a strong proof of security. Loosely speaking, without an explicitly distinguishing query two encrypted tuples are indistinguishable.

There are many open questions in the area of searchable encryption. In the case of outsourced data analytics it is most interesting to combine the efficiency improvements possible for range queries with the necessary security requirements, e.g. by pairing-based cryptography.

9. REFERENCES

- [1] Codification of Auditing Standards and Procedures. Statement on Auditing Standards Number 1. *The American Institute of Certified Public Accountants (AICPA)*. 1972.
- [2] Consideration of Fraud in a Financial Statement Audit. Statement on Auditing Standards Number 99. *The American Institute of Certified Public Accountants (AICPA)*. 2002.
- [3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004.
- [4] D. Boneh, G. DiCrescenzo, R. Ostrovsky, and G. Persiano. Public-key Encryption with Keyword Search. *Proceedings of Eurocrypt*, 2004.
- [5] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. Skeith. Public Key Encryption That Allows PIR Queries. *Proceedings of CRYPTO*, 2007.
- [6] D. Boneh, and B. Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. *Proceedings of Theory of Cryptography Conference*, 2007.
- [7] C. Cachin, S. Micali, and M. Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. *Proceedings of Eurocrypt*, 1999.
- [8] Y. Chang, and M. Mitzenmacher. Privacy Preserving Keyword Searches on Remote Encrypted Data. *Proceedings of 3rd Applied Cryptography and Network Security Conference*, 2005.
- [9] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, 1995.
- [10] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. *Proceedings of ACM Conference on Computer and Communications Security*, 2006.
- [11] S. Evdokimov, and Oliver Günther. Encryption Techniques for Secure Database Outsourcing. *Proceedings of the 12th European Symposium On Research In Computer Security*, 2007.
- [12] E. Goh. Secure Indexes. *Cryptology ePrint Archive: Report 2003/216*. Available at <http://eprint.iacr.org/2003/216/>, 2003.
- [13] O. Goldreich. Secure Multi-party Computation. Available at www.wisdom.weizmann.ac.il/~oded/pp.html, 2002.
- [14] O. Goldreich, and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of ACM* 43(3), 1996.
- [15] P. Golle, B. Waters, and J. Staddon. Secure Conjunctive Keyword Search over Encrypted Data. *Proceedings of the 2nd International Conference on Applied Cryptography and Network Security*, 2004.
- [16] H. Hacigümüs, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over Encrypted Data in the Database-Service-Provider Model. *Proceedings of the 28th ACM SIGMOD Conference on the Management of Data*, 2002.
- [17] B. Hore, S. Mehrotra, and G. Tsudik. A Privacy-Preserving Index for Range Queries. *Proceedings of the 30th International Conference on Very Large Data Bases*, 2004.
- [18] E. Kushilevitz, and R. Ostrovsky. Replication is not needed: Single Database Computationally Private Information Retrieval. *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, 1997.
- [19] A. Lenstra, and E. Verheul. Selecting Cryptographic Key Sizes, *Journal of Cryptology* 14, 2001.
- [20] E. Shi, J. Bethencourt, H. Chan, D. Song, and A. Perrig. Multi-Dimensional Range Query over Encrypted Data. *Proceedings of IEEE Symposium on Security and Privacy*, 2007.
- [21] R. Sion, and B. Carbunar. On the Computational Practicality of Private Information Retrieval. *Proceedings of Network and Distributed System Security Symposium*, 2007.
- [22] Brian Snow. We Need Assurance! *Proceedings of the 21st Annual Computer Security Applications Conference*, 2005
- [23] R. Vanasco. Fraud Auditing. *Managerial Auditing Journal* 13, 1998.
- [24] J. Wells. Billing Schemes, Part 1: Shell companies that don't deliver. *Journal of Accountancy* 194, 2000.
- [25] Z. Yang, S. Zhong, and R. Wright. Privacy-Preserving Queries on Encrypted Data. *Proceedings of the 11th European Symposium On Research In Computer Security*, 2006.

APPENDIX

A. IMPLEMENTATION

We implemented our SEODA scheme in Java using its math library for big integer arithmetic. For simplicity we use one database tuple with one field in our experiments, but our implementation supports arbitrary tuple definitions. The size of this field varies from 4 to 16 bit in increments of 4 bits. In [19] a key size of 139 bits for discrete log based encryption systems is suggested to be safe for the next 3 years. We used key sizes of 160 to 256 bits in increments of 32 bits, such that our encryption system is safe for the future. We replaced the random permutation with the cryptographic hash function SHA-256 which would require a random oracle in the proofs.

We measured running time of encryption, keyword search and range query. We also measured the ciphertext size in

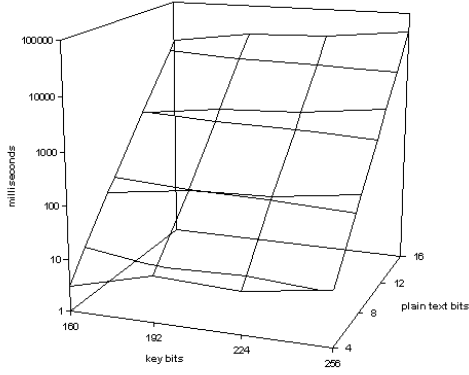


Figure 2: Running Time of Encryption

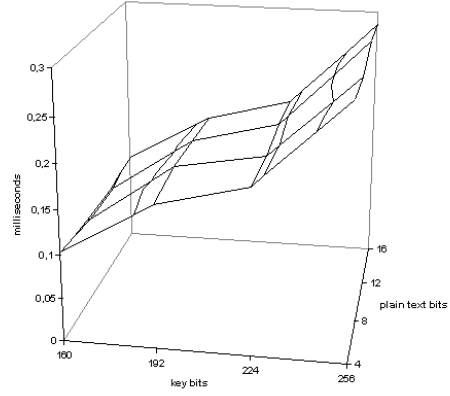


Figure 4: Running Time of Keyword Search

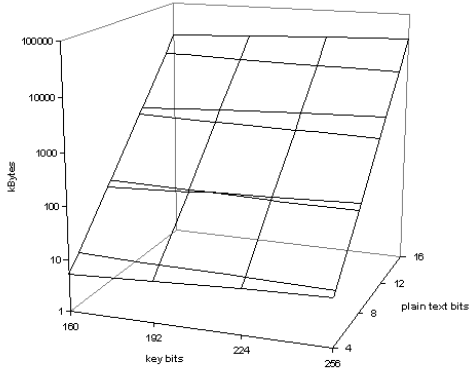


Figure 3: Ciphertext Size

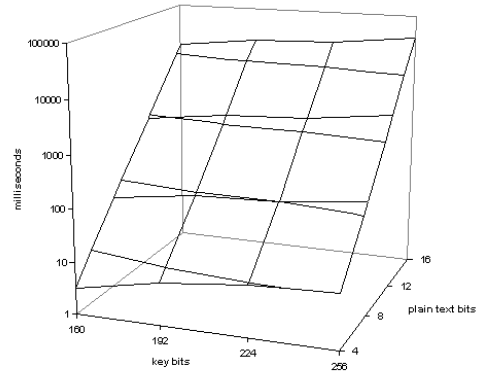


Figure 5: Running Time of Range Query

all cases. The results are depicted in Figures 2– 5. Note the logarithmic scale on the z-axis for encryption running time, ciphertext size, and range query running time. Encryption and range query each correspond to the ciphertext size which is unfortunately exponential in the plaintext size. Encryption performs one modular exponentiation and multiplication per size of the ciphertext ($\mathbb{D}|t|^2$) and one modular exponentiation per domain element (\mathbb{D}). Since $|t|^2 = 1$ in our experiments, encryption performs two modular exponentiations per domain element ($2\mathbb{D}$). Range queries perform $2\mathbb{D}|t|$ modular exponentiations and multiplications per tuple. They decrypt each domain element and check for a permutation (or hash function) match which requires on average $\frac{1}{2}\mathbb{D}|t|$ permutation (or hash function) computations which is negligible in absolute running time compared to the modular exponentiations. The increase in encryption and range query time is steeper than the ciphertext size along the x-axis (key size). This is supported by the cubic complexity of modular exponentiation. Keyword search running time is independent of plaintext size, since always a constant number of values is searched for each query. It requires $|t|$ modular exponentiations and multiplications per tuple. Therefore there is no increase along the y-axis. The same (cubic) increase on the x-axis (key size) is visible for

keyword search running time as for range query and encryption running time.

While running times of 3 ms to 14 s seems acceptable on current hardware (Intel Core Duo 2GHz, 2GB RAM), the ciphertext size of 20 MB for a 16-bit value seems unacceptable for database encryption. In conclusion, the encryption scheme is currently only practical for small domains of less than 16 bit.