Searchable Encryption for Outsourced Data Analytics

Florian Kerschbaum[1] and Alessandro Sorniotti[2]

SAP Research, Karlsruhe, Germany[1]
florian.kerschbaum@sap.com
SAP Research and Institut Eurécom, Sophia Antipolis, France[2]
alessandro.sorniotti@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.

In this paper we present an efficient construction for a searchable encryption scheme for outsourcing data analytics. In this 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 scheme extends previous work by the ability to re-use query results as search tokens (query-result reusability) and the ability to perform range queries. It is efficient with $O(\log^2 n)$ work for a range query and is semantically secure relying only on Diffie-Hellman assumptions (in the random oracle model).

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.

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 data whose access is granted to the auditor 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.

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, so 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.

2 Query Language and Setup

2.1 Query Language

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

Let $t = \langle d^1, d^2, \ldots, d^n \rangle$ 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 for a query expression e is as follows:

$\texttt{e := e} \wedge \texttt{e | e} \vee \texttt{e | s}$
$\texttt{s := t.d}^{\texttt{i}}\texttt{ op c}^{\texttt{i}}\texttt{ | t.d}^{\texttt{i}}\texttt{ ops t}'\texttt{.d}^{\texttt{i}}$
$\texttt{op := ops | < | >}$
$\texttt{ops := ==}$

This grammar implies range queries $(c^i < t.d^i \wedge t.d^i < c'^i(c^i < c'^i))$ and keyword searches $(c^i == t.d^i)$. We write $e \models t$, if e matches the tuple t. The footprint $(\mathbb{C}_{range}, \mathbb{C}_{id}, \mathbb{F})$ of an expression e is its set $\mathbb{C}_{range}$ of used constants in range queries, its set $\mathbb{C}_{id}$ of used constants in identity queries and its set $\mathbb{F}$ 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 $(\{2, 6\}, \{4\}, \{t'.d^3\})$. Note that the query language only represents the queries possible by the data analyst, not the security guarantee enforced by the system which we detail in Section 3.2 using a game-based definition.

2.2 Query Result Reusability

Our query language includes equality comparison of data items of two tuples; given any two encrypted tuples, one can always check whether the two are equal or not, without the need of decrypting them. This clearly implies that if a query has returned a set of tuples, each tuple in the result set can be used in turn as an identity query token: this allows subsequent keyword searches. We call this feature query result reusability.

We could go one step further and require that query results be not only reusable in equality queries but also in range queries. However we refrain from it as this would conflict with the requirements for ciphertext indistinguishability. Indeed, if this were possible, then one 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 $t.d^i < t'.d^i$, consequently breaking any IND-CPA-like game.

A crucial 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.

2.3 Improvements over Previous Work

Our security requirements are identical to public key encryption with oblivious keyword search introduced by [8]. There is Alice who does not want to reveal its query and Bob who does not want to reveal its database. We stress that we considered this setup and developed our solution independently concurrently to [8]. Our work differs in that it is use-case driven from outsourced auditing. Consequently we introduce new functionality, namely the following two features: query result reusability and range queries.

While query result reusability is a novel concept, range queries on encrypted data have been considered in [18] and [6]. We borrow some techniques for range queries from [18], but their scheme does not lend itself to efficiently implementing blind IBE, since it reveals the plaintext in a matching query (match-revealing). Hiding the query from both, the trusted key authority and the database, is a prerequisite for our application. Their ciphertext size is logarithmic in the size of the domain $\mathbb{D}$ of the plaintext. The Boneh-Waters [6] encryption scheme supports queries for arbitrary subsets and opposed to Shi et al. hides the resulting data item in a matching query (match-concealing). It is therefore better suited for query privacy, but still a query token may reveal the data item queried for. Their ciphertext size is the full size of the range: $O(\mathbb{D})$.

Another competing approach is to use private information retrieval (PIR) [7, 10, 17] over single-key encrypted data. We improve over those techniques by reducing the computation complexity to polylogarithmic for range queries and the communication complexity by a factor linear in the number of queries. We know from [19] 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 (but less than in PIR) 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.

Other secure computation protocols, such as private set intersection or Yao's millionaire's protocols are not suitable. While they perform the same functions as our searchable encryption scheme enables, the fundamental problem is that in (almost) any secure computation protocol the function is public, i.e. known to both parties. The entire point of our construction is to have one party chose the function and hide it from the other party.

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 $O(\log(\mathbb{D})|t|l)$, range query token size $O(\log(\mathbb{D}))$, identity query token size $O(1)$, encryption complexity $O(\log(\mathbb{D})|t|l)$, range query complexity $O(\log^2(\mathbb{D})l)$ and keyword search complexity $O(l)$.

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 key K_{DO} at the data owner and a public security parameter P.*
2. **Encrypt**(K_{DO}, t)**:** *Takes a secret key K_{DO} and a tuple t (adhering to Γ) and outputs a ciphertext C.*
3. **PrepareRangeQuery**$[(c^i_{range}, c'^i_{range}), (K_{DO})]$**:** *Is a protocol between the data analyst DA and the data owner DO. The analyst inputs a range from c^i_{range} to c'^i_{range} and the owner inputs a secret key K_{DO}. The output at the analyst is a range query token Q_{range} and the data owner receives no output. The protocol hides the inputs, such that the analyst will learn nothing about K_{DO} and the data owner nothing about c^i_{range}, and c'^i_{range}.*
4. **PrepareIdentityQuery**$[(c^i_{id}), (K_{DO})]$**:** *Is a protocol between the data analyst DA and the data owner DO. The analyst inputs a constant c^i_{id} and the owner inputs a secret key K_{DO}. The output at the analyst is an identity query token Q_{id} and the data owner receives no output. The protocol hides the inputs, such that the analyst will learn nothing about K_{DO} and the data owner nothing about c^i_{id}.*
5. **Analyze**$(C, \mathbb{Q}_{range}, \mathbb{Q}_{id}, e)$**:** *Takes a ciphertext C, a set of range query tokens $\mathbb{Q}_{range}$, a set of identity query tokens $\mathbb{Q}_{id}$ and a query expression e and outputs a set $\mathbb{Q}'_{id}$ of identity query tokens.*

For the encryption scheme to be searchable we impose the following consistency constraint. For each tuple $t = \langle c^1, c^2, \ldots, c^n \rangle$ (defined by Γ) and each query expression e with footprint ($\mathbb{C}_{range}$, $\mathbb{C}_{id}$, $\mathbb{F}$), the above scheme must satisfy the consistency constraint from Figure 1. It basically states that the output of the Analyze algorithm is a set of identity query tokens for each value of the tuples matching the query, or the empty set if none does.

$$\text{Analyze}(C, \mathbb{Q}_{range}, \mathbb{Q}_{id}, e) = \begin{cases} \{\text{PrepareIdentityQuery}[(c), (K_{DO})] | \forall c \in \{c^1, c^2, \ldots, c^n\}\} \\ \hfill \text{if} e \models t \\ \bot \hfill \text{w.h.p., otherwise} \end{cases}$$

where

$$P, K_{DO} = \text{Setup}(k, \Gamma)$$

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

$$\mathbb{Q}_{range} \supset \{\text{PrepareRangeQuery}[(c, c'), (K_{DO})] | c, c' \in \mathbb{C}_{range}\}$$

$$\mathbb{Q}_{id} \supset \{\text{PrepareIdentityQuery}[(c), (K_{DO})] | c \in \mathbb{C}_{id} \cup \mathbb{F}\}$$

Fig. 1. Consistency Constraint

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 $Game_{DA}$ defined below.*

- **Setup:** The data owner runs Setup(k, Γ) and passes the public parameter P to the data analyst (presumed adversary).
- **Query Phase 1:** The data analyst adaptively outputs a (mixed) sequence of either
  - a plain text tuple $t_1, t_2, \ldots t_{q_1}$,
  - a (non-composite) range query constant $(c_1^i, c_1'^i), (c_2^i, c_2'^i), \ldots, (c_{q2}^i, c_{q2}'^i)$, or
  - an identity query constant $c_1^i, c_2^i, \ldots, c_{q3}^i$.

  where q_1, q_2 and q_3 represent an upper bound on the number of encryption, identity and range queries that the data analyst makes. The data owner responds corresponding to the type of the query with either
  - the ciphertext $C = \text{Encrypt}(K_{DO}, t)$.
  - the range query token $Q_{range} = \text{PrepareRangeQuery}\ [(c^i, c'^i), (K_{DO})]$.
  - the identity query token $Q_{id} = \text{PrepareIdentityQuery}\ [(c^i), (K_{DO})]$.
- **Challenge:** The data analyst outputs two different plain-text tuples t_0^* and t_1^* subject to the following restrictions

- either no identity query constant c^i matches the challenge plain texts $t^*_{\{0,1\}}$ or it matches both challenge plain texts in the same dimension, i.e.

$$\begin{aligned}&\forall j \in [1, q_3] \;\; s.t. \\ &(c^i_j \neq t^*_0.c_i \wedge c^i_j \neq t^*_1.c_i) \vee (c^i_j = t^*_0.c_i \wedge c^i_j = t^*_1.c_i)\end{aligned}$$

- no range query (c^i_j, c'^i_j) can distinguish the challenge plain texts $t^*_{\{0,1\}}$, i.e.

$$\begin{aligned}&\forall j \in [1, q_2] \;\; s.t. \\ &((c^i_j > t^*_0.c^i \wedge c^i_j > t^*_1.c^i) \vee (c^i_j < t^*_0.c^i \wedge c^i_j < t^*_1.c^i)) \wedge \\ &((c'^i_j > t^*_0.c^i \wedge c'^i_j > t^*_1.c^i) \vee (c'^i_j < t^*_0.c^i \wedge c'^i_j < t^*_1.c^i))\end{aligned}$$

- none of t_j's constants $t_j.c^i$ matches any constant of the challenge plain texts $t^*_{\{0,1\}}$, i.e.

$$\forall j \in [1, q_1], \forall i \;\; s.t. \;\; t.c^i \neq t^*_0.c^i \wedge t.c^i \neq t^*_1.c^i$$

The data owner flips a coin $b \in \{0, 1\}$ and encrypts t^*_b under K_{DO}. The ciphertext $C^* = \text{Encrypt}(K_{DO}, t^*_b)$ is passed to the data analyst.

- **Query Phase 2:** The data analyst continues to adaptively output plaintexts, range query and identity query constants subject to the restrictions above. The data owner responds with the corresponding ciphertexts, range and identity query tokens.
- **Guess:** The data analyst outputs a guess b' of b.

A data analyst $\mathcal{A}$'s advantage in the above game is defined as

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

We need to exclude identity constant queries that match only one challenge plaintext, because they could be used in a query to distinguish the ciphertext. For the same reason we need to exclude range queries that can distinguish ciphertexts. We need to exclude encryptions of any challenge ciphertext in any scheme, since they could be decrypted to identity query tokens by matching range queries which could then distinguish the challenges.

4 Building Blocks

Let us first introduce terminology that will be used in the rest of this paper. In what follows, we denote $\mathbb{Z}^*_p = \{1, \ldots, p - 1\}$.

Given a security parameter k, let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be groups of order p for some large prime p, where the bit-size of p is determined by the security parameter k. Our scheme uses a computable, non-degenerate bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$

for which the *Symmetric External Diffie-Hellman (SXDH)* problem is assumed to be hard. The SXDH assumption in short allows for the existence of a bilinear pairing, but assumes that the Decisional Diffie-Hellman problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$ and was used e.g. in [1].

We recall that a bilinear map satisfies the following three properties:

- Bilinear: for $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$ and for $a, b \in \mathbb{Z}_p^*$
$$\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$$
- Non-degenerate: $\hat{e}(g, h) \neq 1$ is a generator of $\mathbb{G}_T$
- Computable: there exists an efficient algorithm to compute $\hat{e}(g, h)$ for all $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$

4.1 Identity-based Encryption

We capitalize from Waters' [20] and Boneh Boyen Goh's [2] IBE scheme, with some differences. First of all, we modify the scheme so as to include the SXDH assumption, discussed previously. Secondly, we do not adopt Waters' nice hashing scheme but we require random oracles for reasons of our proofs and therefore we use standard hash functions that map strings onto group elements. Note that the random oracle, i.e. the hash function, only operates in one of the two groups of the SXDH assumption.

The IBE scheme has the following algorithms:
Setup(k): Let H be a one way hash function defined from $\{0, 1\}^*$ to $\mathbb{G}_2$. The public parameters are $g \in \mathbb{G}_1$, g^α, $h \in \mathbb{G}_2$. Messages need to be encoded in group $\mathbb{G}_T$ The secret parameter is α. We denote the identity id.
Encrypt(id, m): Choose $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$.

$$C = \langle g^s, H(id)^s, \hat{e}(g^\alpha, h)^s m \rangle$$

GetPrivateKey(id, α): Choose $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$.

$$k_{id} = \langle g^r, h^\alpha H(id)^r \rangle$$

Decrypt(C, k_{id}):

$$\hat{e}(g^\alpha, h)^s m \frac{\hat{e}(g^r, H(id)^s)}{\hat{e}(g^s, h^\alpha H(id)^r)} = m$$

5 Our SEODA Scheme

For simplicity of the exposition we construct a SEODA scheme for 1-dimensional tuples $t = \langle c \rangle$ in this section. Multi-dimensional tuple definitions are covered in an extended version of the paper.

Let $\mathbb{D} = [d_1, d_2]$ be the domain of c. Let us first explain how we represent ranges in $\mathbb{D}$. We organize all the elements of $\mathbb{D}$ in ascending order as the leaves

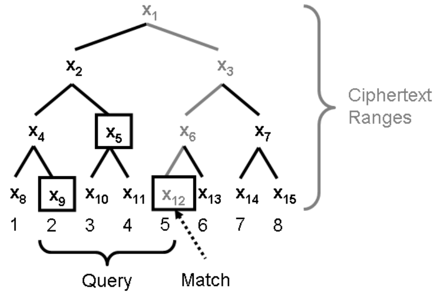

Fig. 2. Representing ranges on a binary tree

of a binary tree. Figure 2 shows the tree that is created when $\mathbb{D} = [1, 8]$. Each element is labeled with an identity. This way we have identified $O(|\mathbb{D}|)$ intervals, where each node defines an interval comprised of all the elements of $\mathbb{D}$ that are the leaves of the subtree rooted at the node itself. For instance, with reference to Figure 2, x_2 identifies the interval $[1, 4]$. With combinations of such intervals, we can identify any range in the domain $\mathbb{D}$. For instance the interval $[2, 5]$ is identified by the union of x_9, x_5 and x_{12}. We require the data analyst to query each interval in a separate protocol and later compose the result with an $\wedge$ join.

With this in mind, let us see how we build our scheme.

Setup(k, Γ):

The data owner (DO) sets up the IBE scheme defined in the previous Section. DO also picks $t_{DO} \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and publishes $h^{t_{DO}}$. Finally, DO creates a binary tree $T_{\mathbb{D}}$ for the domain $\mathbb{D}$ and makes the identifiers of each node public.

Encrypt(K_{DO}, t):

DO picks $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and computes the identity token

$$ID_t = \langle g^s, H(t)^{s t_{DO}} \rangle$$

DO then selects from $T_{\mathbb{D}}$ the $O(\log n)$ identities $X_t = \{x_i$: node i is in the path from t to the root$\}$ for all ranges from the leaf corresponding to t up to the top of the tree. With reference to Figure 2 once more, if $t = 5$, the considered identities would be x_{12}, x_6, x_3 and x_1. Note that in case of a match between range query and plaintext, there is one and only one range (identity) in common between query and ciphertext. Then, DO IBE-encrypts ID_t under all the identities in X_t. Since ID_t's encoding in $\mathbb{G}_T$ is longer than a single group element, the data owner DO should pick a fresh random key k for a symmetric encryption scheme and encrypt k using the identities. DO can then encrypt ID_t and a checksum for integrity verification using this key k in the symmetric cipher.

After the encryption, DO returns ID_t along with its $\log n$ IBE encryptions.

PrepareIdentityQuery[(c_{id}), (K_{DO})]:

DA wants to request an identity token for a value $c_{id} \in \mathbb{D}$. He picks $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$, and sends $H(c_{id})^r$ to DO.

DO picks $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and replies with $(H(c_{id})^r)^{st_{DO}}$, g^s.

DA computes $(H(c_{id})^{rst_{DO}})^{r^{-1}}$, thus obtaining the identity token for c_{id}

$$ID_{c_{id}} = \langle g^s, H(c_{id})^{st_{DO}} \rangle$$

PrepareRangeQuery[(c_{range}, c'_{range}), (K_{DO})]**:**

The data analyst DA wants to obtain a range query token for a single range $[c_{range}, c'_{range}]$ within the binary tree of ranges. DA consequently chooses the identity x_r that represents such a range. DA chooses one $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and sends $H(x_r)^r$ and h^r to the data owner DO. We emphasize that both, $H(x_r)^r$ and h^r, are elements of $\mathbb{G}_2$ and due to the SXDH assumption no efficient linear map $f() : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ can exist, such that computing $\hat{e}\left(f(h^r), H(x_r)\right) = \hat{e}\left(f(h), H(x_r)^r\right)$ is infeasible.

After receiving the identity, DO picks $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$. Then DO returns to DA

$$\langle g^s, h^{r\alpha} H(x_r)^{rs} \rangle$$

Upon receipt, DA raises the first term to the multiplicative inverse of r, thus obtaining the following IBE decryption key for identity $H(x_r)$:

$$k_{x_r} = \langle g^s, h^{\alpha} H(x_r)^s \rangle$$

The resulting complexity for a single range is $O(\log \mathbb{D})$. For complex range queries DA must request each range individually and combine the results by himself. Note that there are at most $O(\log \mathbb{D})$ ranges that need to be combined resulting in a complex range query complexity $O(\log^2 \mathbb{D})$.

Analyze(C, $\mathbb{Q}_{range}$, $\mathbb{Q}_{id}$, e**):**

For range queries, DA decrypts each IBE encryption of ID_t with each IBE decryption key received from DO upon his query. If any combination decrypts to ID_t, the match was successful.

For identity queries, DA owns (a set of) ID_y for some queried value y. He can check for equality of any tuple t, disposing of $ID_t = \langle g^s, H(t)^{st_{DO}} \rangle$ and $ID_y =< H(y)^{rt_{DO}}, g^r >$ by checking whether

$$\hat{e}\left(g^r, H(t)^{st_{DO}}\right) = \hat{e}\left(g^s, H(y)^{rt_{DO}}\right)$$

holds. If it does, DA concludes that the match was successful.

6 Security

6.1 Symmetric External Diffie Hellman Assumption

Definition 3 *We say that the SXDH assumption holds if, given values $y, y_1, y_2, y_3 \in \mathbb{G}_1$, it is not computationally feasible to decide if there is an integer $a \in \mathbb{Z}_p^*$ such that $y_1 = y^a$ and $y_3 = y_2^a$, i.e., $\mathbb{G}_1$ is a DDH-hard group. The same requirement must hold for $\mathbb{G}_2$, i.e., it is also a DDH-hard group.*

6.2 Bilinear Decisional Diffie Hellman Assumption

Definition 4 *We say that the BDDH assumption holds if, given values $g, g^a, \in \mathbb{G}_1$, $h, h^b, h^c \in \mathbb{G}_2$, $\hat{e}(g,h)^x \in \mathbb{G}_T$ it is not computationally feasible to decide if $x = abc$.*

6.3 Protocol Security

We define security of the protocols for PrepareRangeQuery and PrepareIdentityQuery as semi-honest security for secure two-party computation protocols [14], 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*

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

where r represents the outcome of X'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 $VIEW_X^{\Psi}$:*

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

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 Theorems 1 and 2 by the simulators $S_{PrepareRangeQuery}$ and $S_{PrepareIdentityQuery}$.

For the data owner's view simulator $S_{PrepareRangeQuery}$ outputs 2 uniform random elements from $\mathbb{G}_2$. Simulator $S_{PrepareIdentityQuery}$ outputs 1 uniform random elements from $\mathbb{G}_2$ for its data owner's view.

Furthermore we need to show that the views are indeed computationally indistinguishable. First, note that the view of the data owner in the PrepareRangeQuery protocol is clearly a superset of its view in the PrepareIdentityQuery protocol. We therefore conclude that if an algorithm is unable to distinguish the view in the PrepareRangeQuery protocol, it is unable to do so in the PrepareIdentityQuery protocol.

We define a game $Game_{DO}$ for an adversary acting as the data owner. In this game the data owner is given one range query and then asked to tell whether it corresponds to a valid range or it is chosen randomly as by the simulator. The security defined by Game $Game_{DO}$ even holds for a binary domain, i.e. there are only two possible values for the range.

We modify the setup of the encryption scheme in the setup of $Game_{DO}$, such that the challenger, i.e. the data analyst, gets to pick the groups the operations are performed in. Note that this does not break the security of the encryption scheme, since the data owner DO can still choose its secret key t_{DO} and the secret parameter of the IBE scheme, as long as the SXDH assumption holds. A query phase in $Game_{DO}$ has been omitted, since the input in a real attack is entirely under control of the data analyst.

$Game_{DO}$ is defined as follows:

- **Setup:** The simulator chooses the initial public parameter P' of Setup(k, Γ) and passes it to the data owner. The data owner completes Setup(k, Γ) by choosing the secret keys and passes the dependent public parameter P'' to the simulator ($P = P' \cup P''$).
- **Challenge:** The simulator sends the data owner one range query request. Note that our reduction would still work, if the simulator also passes the corresponding plaintext range x, which underpins our security against known plaintext-like attacks. The simulator challenges the data owner to tell whether the request is valid or randomly chosen numbers.
- **Guess:** The data owner outputs a guess b ($b = 0$ for a valid request, $b = 1$ for randomly chosen numbers).

A data owner $\mathcal{A}$'s advantage in the above game is defined as

$$Adv_{\mathcal{A}} = |Pr[\mathcal{A}[b] - \frac{1}{2}|$$

Lemma 1 *Suppose there is a data owner $\mathcal{A}$ that has an advantage ϵ in breaking game Game$_{DO}$. Then there exists an algorithm $\mathcal{B}$ that solves DDH in $\mathbb{G}_2$ with advantage at least:*

$$Adv_{\mathcal{B}} \geq \frac{1}{2} + \epsilon$$

Its running is $O(time(\mathcal{A}))$.

A proof of Lemma 1 can be found in an extended version of the paper.

It remains to show that the data analyst's view in the protocols PrepareRangeQuery and PrepareIdentityQuery can be simulated by $S_{PrepareRangeQuery}$ and $S_{PrepareIdentityQuery}$, respectively. In fact this is simple, since in both cases the view is identical to the output. The simulators which have access to the output can therefore simulate the data analyst's views by simply copying the output.

6.4 Ciphertext Indistinguishability

Theorem 3 *Suppose the hash function H is a random oracle. Then an attacker $\mathcal{A}$ has a negligible advantage in winning the security game $Game_{DA}$ assuming the BDDH assumption holds.*

We prove Theorem 3 by reducing an attacker in game $Game_{DA}$ to an attacker of the BDDH challenge.

Lemma 2 *Suppose there is a data analyst $\mathcal{A}$ that has an advantage ϵ in breaking game $Game_{DA}$. Suppose $\mathcal{A}$ makes at most q_H hash queries to H, at most q_E encryption requests and engages in at most q_I PrepareIdentityQuery protocols. Then there exists an algorithm $\mathcal{B}$ that solves BDDH with advantage at least:*

$$Adv_{\mathcal{B}} \geq \frac{\epsilon}{2e(1 + q_E + q_I)}$$

Its running is $O(time(\mathcal{A}) + (q_H + q_E)q_I)$.

The proof of Lemma 2 is in Appendix A of this paper. Its main idea is adapted from [4].

7 Related Work

A first SEODA scheme has been presented in [16], but it neither enjoyed semantic security relying on the discrete logarithm assumption which does not result in bit security nor was it practically efficient with an encryption time of $O(\mathbb{D}|t|^2)$ per tuple. The scheme in this paper enjoys stronger security relying only on Diffie-Hellman assumptions and reduces the time for range queries to $O(\log^2 \mathbb{D})$ per tuple.

Other examples of such searchable encryption schemes are [9, 11–13, 21]. 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 [15].

Public-key encrypted, oblivious, keyword search was introduced in [8]. We use the same notion of obliviousness (i.e. privacy of the query), but extend by range queries and query-result reusability. Our construction is more efficient and the generation of the public parameters is significantly simplified, since we do not need to combine homomorphic encryption and bilinear maps.

Keyword searches are important, but to be useful in practice, range queries are indispensable. The problem of range queries has been addressed in [6, 18].

Searchable encryption with range queries is presented in [6, 18]. 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 [18] where the database owner publishes his data, but only gives decryption keys (for certain ranges) to qualified users. An example is log data for network traceback.

The first schemes to extend searchable encryption to public key encryption are [3, 5]. 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 [3] and private index queries are described in [5].

Private information retrieval (PIR) [7, 10, 17] 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. Using PIR on encrypted data is significantly less efficient than our approach. It requires processing each tuple for each query on the service providers' side which is less practical than transferring the entire database [19].

8 Conclusion

We considered the problem of outsourcing data analytics, a special case of outsourced auditing. In a such scenario the privacy requirements of data analyst and data owner must be fulfilled and neither the data nor the queries may be revealed.

We present an efficient solution with polylogarithmic range query and polynomial encryption time. We allow range and identity queries and results of those queries can be re-used in subsequent queries as identity query tokens. We proved our scheme and its associated protocols secure under the Bilinear Decisional Diffie-Hellman and the Symmetric External Diffie-Hellman assumption.

Acknowledgements

We would like to thank Julien Vayssiere for initializing discussions on the topic and the reviewers for their insightful comments. This work has been partially financed by the European Commission through the ICT Programme under the Seventh Framework Programme IST Project "Secure Supply Chain Management" (SecureSCM), grant agreement number 213531.

References

1. G. Ateniese, M. Blanton, and J. Kirsch. Secret Handshakes with Dynamic and Fuzzy Matching. *Proceedings of Network and Distributed System Security Sympo-suim*, 2007.

2. D. Boneh, E. Goh, and X. Boyen. Hierarchical Identity Based Encryption with Constant Size Ciphertext. *Proceedings of Eurocrypt*, 2005.
3. D. Boneh, G. DiCrescenzo, R. Ostrovsky, and G. Persiano. Public-key Encryption with Keyword Search. *Proceedings of Eurocrypt*, 2004.
4. D. Boneh, and M. Franklin. Identity-based Encryption from the Weil Paring. *SIAM Journal of Computing 32(3)*, 2003.
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. J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy. Blind and Anonymous Identity-Based Encryption and Authorised Private Searches on Public Key Encrypted Data. *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography*, 2009.
9. Y. Chang, and M. Mitzenmacher. Privacy Preserving Keyword Searches on Remote Encrypted Data. *Proceedings of 3rd Applied Cryptography and Network Security Conference*, 2005.
10. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, 1995.
11. 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.
12. S. Evdokimov, and Oliver Günther. Encryption Techniques for Secure Database Outsourcing. *Proceedings of the 12th European Symposium On Research In Computer Security*, 2007.
13. E. Goh. Secure Indexes. *Cryptology ePrint Archive: Report 2003/216*. Available at http://eprint.iacr.org/2003/216/, 2003.
14. O. Goldreich. Secure Multi-party Computation. Available at *www.wisdom.weizmann.ac.il/˜oded/pp.html*, 2002.
15. O. Goldreich, and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of ACM 43(3)*, 1996.
16. F. Kerschbaum, and J. Vayssiere. Privacy-Preserving Data Analytics as an Outsourced Service. *Proceedings of the ACM Secure Web Services Workshop*, 2008.
17. 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.
18. 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.
19. R. Sion, and B. Carbunar. On the Computational Practicality of Private Information Retrieval. *Proceedings of Network and Distributed System Security Symposium*, 2007.
20. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. *Proceedings of Eurocrypt*, 2005.
21. 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.

A Proof of Lemma 2

Proof. We show how to construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage $\frac{\epsilon}{2e(q_E+q_I+1)}$ against our security game. Algorithm $\mathcal{B}$ is given an instance $A = g^a, B = h^b, C = h^c, Z$ of the BDDH problem. It needs to output a guess whether $Z = h^{abc}$.

Note, that we chose challenge elements from $\mathbb{G}_1$ and $\mathbb{G}_2$, but even given an efficient isomorphism $\xi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$ our problem remains hard and we even ruled out the existence of such an isomorphism by the SXDH assumption.

The challenge in constructing $\mathcal{B}$ is to be able to answer to the secret Prepare-IdentityQuery protocol request. We do this by using the information in the random oracle and therefore answer H queries as follows. In this proof the random oracle is under the control of the data owner (albeit also algorithm $\mathcal{B}$). We adopted this idea from the proof of the Boneh-Franklin identity-based encryption system [4]:

H queries: At any time algorithm $\mathcal{A}$ can query the random oracle H. In order to respond to these queries the algorithm $\mathcal{B}$ maintains a list of tuples $\langle x, hash, y, coin\rangle$ as explained below. We refer to this list as the $H_{DO} - list$. The list is initially empty. When $\mathcal{A}$ queries the oracle H with a bit-sequence $x \in \mathbb{D}$ algorithm $\mathcal{B}$ responds as follows.

1. If the query sequence x appears on the $H_{DO}-list$ in a tuple $\langle x, hash, y, coin\rangle$, then algorithm $\mathcal{B}$ responds with $H(x) = hash$.
2. Otherwise, $\mathcal{B}$ flips a random coin $coin' \in \{0, 1\}$ so that $Pr[coin' = 0] = \delta$ for some δ that will be determined later.
3. Algorithm $\mathcal{B}$ picks a random y' in $\mathbb{Z}_p^*$. If $coin' = 0$ $\mathcal{B}$ computes $hash' = h^{y'}$, else if $coin' = 1$ $\mathcal{B}$ sets $hash' = B$.
4. Algorithm $\mathcal{B}$ adds the tuple $\langle x, hash', y', coin'\rangle$ to the $H_{DO} - list$ and responds to $\mathcal{A}$ with $H(x) = hash'$. Note that $hash'$ is uniform in $\mathbb{G}_2$ and independent of $\mathcal{A}$'s current view.

Setup: Algorithm $\mathcal{B}$ creates the IBE scheme and gives $\mathcal{A}$ the following parameters $\mathbb{G}_1$, $\mathbb{G}_2$, g, g^α, h, H. Let H be a random oracle controlled by $\mathcal{B}$ as described above. Furthermore $\mathcal{B}$ sets $h^{t_{DO}} = C$.

Query Phase 1: Algorithm $\mathcal{A}$ may now send plaintexts, range query requests and identity query requests. We show how $\mathcal{B}$ answers those.

Encryption Requests: Algorithm $\mathcal{A}$ sends plaintext t. $\mathcal{B}$ invokes the random oracle H. It retrieves the tuple $\langle t, hash, y, coin\rangle$ from the $H_{DO} - list$. If the coin flip $coin$ is 1, then $\mathcal{B}$ aborts. We now know that $coin = 0$ and therefore $H(t) = h^y$. $\mathcal{B}$ chooses $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and computes

$$ID_t = C^{sy} = H(t)^{st_{DO}}$$

It sends ID_t, g^s to algorithm $\mathcal{A}$.

Range Query Requests: Algorithm $\mathcal{A}$ chooses a range and its corresponding identity x_r. It also chooses a random number r and sends $H(x_r)^r$ and h^r to algorithm $\mathcal{B}$.

$\mathcal{B}$ chooses a random number $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$. It returns $h^{r\alpha}H(x_r)^{rs}, g^s$.

Identity Query Requests: Algorithm $\mathcal{A}$ chooses the value t, chooses $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and sends $H(t)^r, h^r$ to $\mathcal{B}$. Note, that $\mathcal{A}$ must have invoked H for t.

Algorithm $\mathcal{B}$ retrieves all tuples $t, hash, y, coin$ from the $H_{DO} - list$ where $coin = 0$. For each tuple it checks whether $(h^r)^y = H(t)^r$. If it finds such a tuple, it takes note of y, else if it does not find such a tuple, $\mathcal{B}$ aborts. We now know again that $coin = 0$ and that $H(t) = h^y$. $\mathcal{B}$ picks $s \overset{R}{\leftarrow} ZZ$ and returns $C^{sy} = H(t)^{st_{DO}}$, g^s.

Challenge: Once algorithm $\mathcal{A}$ decides that the first phase is over it sends two plaintexts $t_0^\star$ and $t_1^\star$ to $\mathcal{B}$.

$\mathcal{B}$ flips a random coin b and chooses $t_b^\star$. It invokes the random oracle H with $t_b^\star$ and retrieves its tuple $t_b^\star, hash, y, coin$ from the $H_{DO} - list$. If $coin = 0$, $\mathcal{B}$ reports failure and aborts. We now know that $coin = 1$ and $H(t) = B$.

$\mathcal{B}$ sets $ID_t = Z$ and $g^s = A$. Note that, since $h^{t_{DO}} = C$, ID_t is a valid ciphertext for t if $Z = h^{abc}$.

It then furthermore selects the identities $x_1, \ldots, x_l$ in T_B according to $t_b^\star$. It encrypts ID_t under each identity x_i. Finally it returns the IBE encryptions and ID_t to $\mathcal{A}$.

Query Phase 2: Algorithm $\mathcal{B}$ responds to $\mathcal{A}$'s queries as in query phase 1.

Guess: Algorithm $\mathcal{A}$ eventually outputs its guess b'. $\mathcal{B}$ outputs b' as its guess.

Claim: If algorithm $\mathcal{B}$ does not abort during the simulation $\mathcal{A}$'s view is identical to its view in a real attack. The responses to H queries are as in a real attack, since each is uniformly and independently distributed in $\mathbb{G}_2$.

According to the rules, no range or identity query can distinguish the challenge plaintext. The only information about plaintexts must stem from the ciphertexts.

If $Z = h^{abc}$, then $\mathcal{A}$ has advantage $Adv_{\mathcal{A}} \geq \epsilon$ in breaking game $Game_{DA}$, since its receives a valid ciphertext. If Z is a random number, then the message part $H(t)^{st_{DO}}$ of the identity query token is randomly distributed in $\mathbb{G}_2$ and contains no information to distinguish $t_0^\star$ and $t_1^\star$. Therefore if $\mathcal{B}$ does not abort, $|Pr[b = b'] - \frac{1}{2}| \geq \frac{1}{2}\epsilon$.

To complete the proof of Lemma 2 we need to calculate the probability that algorithm $\mathcal{B}$ aborts during the simulation. Suppose $\mathcal{A}$ makes q_E encryption requests and q_I identity query token requests. Then the probability that $\mathcal{B}$ does not abort in query phases 1 or 2 is $\delta^{q_E+q_I}$. The probability that it does not abort during the challenge step is $1 - \delta$ which results in an overall probability that $\mathcal{B}$ does not abort is $\delta^{q_E+q_I}(1 - \delta)$. This value is maximized at $\delta_{opt} = 1 - \frac{1}{q_E+q_I+1}$. Using δ_{opt} the probability that $\mathcal{B}$ does not abort is at least $\frac{1}{e(q_E+q_I+1)}$ where e is Euler's constant (the base of the natural logarithm). Then $\mathcal{B}$'s advantage in breaking $BDDH$ is at least $\frac{\epsilon}{2e(q_E+q_I+1)}$.

The running time of algorithm $\mathcal{B}$ is the running time of algorithm $\mathcal{A}$ plus the searches in the $H_{DO} - list$ for identity query token requests. Suppose $\mathcal{A}$ makes q_I identity query requests, then there are at most q_I searches in a $H_{DO} - list$ of length at most $q_H + q_E$. The resulting running time is $O(time(\mathcal{A}) + q_I(q_H + q_E))$.