# An Efficiently Searchable Encrypted Data Structure for Range Queries

Florian Kerschbaum[1] and Anselme Tueno[2]

[1] University of Waterloo, Canada
[2] SAP, Karlsruhe, Germany

**Abstract.** At CCS 2015 Naveed et al. presented first attacks on efficiently searchable encryption, such as deterministic and order-preserving encryption. These plaintext guessing attacks have been further improved in subsequent work, e.g. by Grubbs et al. in 2016. Such cryptanalysis is crucially important to sharpen our understanding of the implications of security models. In this paper we present an order-preserving encryption scheme in the form of an efficiently searchable, encrypted data structure that is provably secure against these and even more powerful chosen plaintext attacks. Our data structure supports logarithmic-time search with linear space complexity. The indices of our data structure can be used to search by standard comparisons and hence allow easy retrofitting to existing database management systems. We implemented our scheme and show that its search time overhead is only 10 milliseconds compared to non-secure search on a database with 1 million entries.

## 1 Introduction

At CCS 2015 Naveed et al. [27] presented attacks on order-preserving encryption. Later Grubbs et al. [16] improved the precision of these attacks. Further attacks on searchable encryption have been presented [7, 12, 14, 18, 20, 23, 29, 32]. Such cryptanalysis is crucially important to sharpen our understanding of the implications of security models, since many of the attacked encryption schemes are proven secure in their specific security models. In this paper we formalize security against these attacks and show a connection to chosen plaintext attacks. We also demonstrate that there exists an encrypted data structure that supports efficient range queries by regular comparisons and that provably prevents these attacks.

Comparison using regular, unmodified comparison operators, e.g., greater-than, as enabled by our scheme and order-preserving encryption has many practical benefits. These encryption schemes can be retrofitted to existing database management systems making them extra-ordinarily fast, flexible and easy-to-deploy. We preserve this property as our implementation demonstrates, but some minor modifications to the search procedure are necessary.

Efficiency – logarithmic time and linear space complexity – is also an important property of search over encrypted data. In Table 1 we provide a comparison of our scheme to the most secure and efficient order-preserving schemes

**Table 1.** Comparison of efficiency and security of schemes for range queries over encrypted data

| Scheme | Search Time $O(\log n)$ | Space $O(n)$ | IND-CPA-DS-secure[3] |
|---|---|---|---|
| Partial order preserving encoding [30] | Yes | Yes | Only before queries |
| Searchable encryption with replicated index [10] | Yes | No | Yes |
| Searchable encryption with dynamic index [17] | Only amortized | Yes | Only before queries |
| Searchable encryption with index replacement [2] | Yes | Yes | No |
| Order-revealing encryption [24] | No | Yes | Yes |
| This paper | Yes | Yes | Yes |

[30], order-revealing encryption [24] and range-searchable encryption [2, 10, 17] schemes. No searchable encryption scheme – including ours – offers perfect security and efficiency for all functions (equality and range search, insertions, deletions, etc.). It is a research challenge to balance the trade-off between the two objectives, even for a restricted set of functions. We aim at provable security against the recently publicized plaintext guessing attacks while still enabling efficient range search. In this respect, we achieve a novel and preferable trade-off between security and efficiency.

The construction of our scheme combines the benefits of previous order-preserving encryption schemes: modular order-preserving encryption by Boldyreva et al. [4], ideal secure order-preserving encoding by Popa et al. [28] and frequency-hiding order-preserving encryption by Kerschbaum [21]. We assign a distinct ciphertext for each – even repeated – plaintext as Kerschbaum does, but his scheme statically leaks the partial insertion order. So, we compress the randomized ciphertexts to the minimal ciphertext space using Popa et al.'s interactive protocol. Then we rotate around a modulus as Boldyreva et al., but on the ciphertexts and not on the plaintexts.

As a result we achieve *structural independence* between the ciphertexts and plaintexts which is a prerequisite for security against chosen plaintext attacks and plaintext guessing attacks – particularly, if the adversary has perfect background knowledge on the distribution of plaintexts. We formalize this insight as a novel security model (IND-CPA-DS-security) for efficiently searchable, encrypted data structures and we prove our scheme secure in this model. Our security model encompasses a number of recently publicized attacks where attackers broke into cloud system and stole the stored data. Such an attack will reveal no additional information when data is encrypted with our scheme. This will also thwart the attacks by Naveed et al. [27] and Grubbs et al. [16] mentioned at the beginning of the introduction.

[3] IND-CPA-DS is defined in Section 3.2.

As a scientific contribution we falsify two hypotheses commonly assumed to be true in the database security community. The first hypothesis is that all order-preserving encryption schemes are susceptible to ciphertext-only attacks – as those demonstrated by Naveed et al. and Grubbs et al. We do this by carefully defining security against such attacks and then constructing a provably secure scheme. The second hypothesis is that in order to securely store values in a database it suffices to encrypt them using a probabilistic encryption scheme (e.g. AES in CBC or GCM mode). The fallacy is ignoring that in order to store the data it has to have an address in the storage medium. If this address correlates with the data, probabilistic encryption is insufficient. Oblivious RAM (ORAM) randomizes this address under dynamic accesses and our security definition does the same for static data.

## 2 Efficiently Searchable Encrypted Data Structures

First, we define what we mean by an efficiently searchable encrypted data structure (ESEDS). We start by defining what we mean by a data structure. We use the fundamental representation of a data structure in random-access memory, i.e. an array. Each cell of the array consists of a structured element. We do not impose any restriction on the structure of the element, but usually this element contains two parts: the data to be searched over and further structural information, such as indices of further entries. Note that structural information may be *implicit*, i.e. the index where an element is stored itself is structural information albeit not explicitly stored. This implicit structural information may also not be encrypted, but only randomized. An example of *explicit* structural information are the indices of the cells of the two children in a binary search tree which would be stored in a cell's structure in addition to the data of a tree node. Explicit structural information can be encrypted. We write $\mathbb{C}[j]$ for the $j$-th element and if it is clear from the context, we assume it consists only of a ciphertext of the data element (with $j$ being the implicit structural information).

**Definition 1** (DS)**.**
*A data structure* DS *consists of an array of elements* $\mathbb{C}[j]$ *(*$0 \leq j < n$*).*

For an encrypted data structure there are a number of options on the type of encryption. First, we can choose symmetric or public-key encryption. We can instantiate our encrypted data structure with either one. Let PSE be a probabilistic symmetric encryption scheme consisting of three – possibly probabilistic – polynomial-time algorithms $\mathsf{PSE} = \mathsf{KGen}(1^\lambda)$, $\mathsf{Enc}(\mathsf{k}, m)$, $\mathsf{Dec}(\mathsf{k}, c)$. Let PPKE be a probabilistic public-key encryption scheme consisting of three – possibly probabilistic – polynomial-time algorithms $\mathsf{PPKE} = \mathsf{KGen}(1^\lambda)$, $\mathsf{Enc}(\mathsf{pk}, m)$, $\mathsf{Dec}(\mathsf{sk}, c)$. Let $\mathsf{pk} \leftarrow \mathsf{KDer}(\mathsf{sk})$ be a deterministic algorithm that derives the public key from the private key in a public-key encryption scheme. For symmetric key encryption let KDer be the identity function. Let $\mathsf{PE} \in \{\mathsf{PSE}, \mathsf{PPKE}\}$ and we use PE when we leave the choice of encryption scheme open.

Second, we can either encrypt the data structure as a whole or parts of the data structure – ideally each cell. Our requirement of efficient search rules out the first option. Since in this case each search operation would require decrypting the data structure which is at least linear in the ciphertext size, sublinear search is impossible. Hence, we require each cell to be encrypted as a separate ciphertext.[4]

Third, for data security it may only be necessary to encrypt the data elements of a cell and not the structural information. In fact, our own proposed ESEDS is an instance of such a case where the structural information is implicit from the array structure and unencrypted. Hence, we only require the data element of each cell to be encrypted.

**Definition 2** (EDS)**.**
*An encrypted data structure* $\mathsf{EDS}_{\mathsf{PE}}$ *consists of an array of elements* $\mathbb{C}[j]$ *where at least the data part has been encrypted with* PE*.*

We can now define the operations on a searchable encrypted data structure $\mathsf{SEDS}_{\mathsf{PE}}$. We write SEDS when the choice encryption scheme is clear from the context. Furthermore we sometimes denote the version $h$ (after $h$ insertions) of a data structure as $\mathsf{SEDS}^h$. Our definition is for range-searchable encrypted data structures, but this implies a definition for keyword searchable data structure as well (where the range parameters are equal: $a = b$). Furthermore, we do not define how operations on our SEDS are to be implemented. These operations can be implemented as algorithms running on a single machine or protocols distributed over a client and server (hiding the secret key from the server). Both choices are covered by our definition.

**Definition 3** (SEDS)**.**
*A searchable encrypted data structure* $\mathsf{SEDS}_{\mathsf{PE}}$ *offers the following operations.*

- $\mathsf{k} \leftarrow \mathsf{KGen}(1^\lambda)$*: Generates a – either secret or private – key* k *from the encryption scheme* PE *according to the security parameter* $\lambda$*.*
- $\mathbb{C}^{h+1} \leftarrow \mathsf{Enc}(\mathsf{k}, \mathbb{C}^h, m)$*: Encrypts the plaintext* $m$ *using* $\mathsf{PE.Enc}(\mathsf{KDer}(\mathsf{k}), m)$ *and inserts it into the data structure* $\mathbb{C}^h$ *resulting in data structure* $\mathbb{C}^{h+1}$*.*[5]
- $m := \mathsf{Dec}(\mathsf{k}, \mathbb{C}[j])$*: Computes the plaintext* $m$ *for the data part of encrypted cell* $\mathbb{C}[j]$ *using key* k*.*
- $\{j_0, \ldots, j_{\ell-1}\} := \mathsf{Search}(\mathsf{k}, \mathbb{C}, a, b)$*: Computes the set of indices* $\{j_0, \ldots, j_{\ell-1}\}$ *for the range* $[a, b]$ *on the encrypted data structure* $\mathbb{C}$ *using key* k*.*

For the correctness of encryption we expect in a sequence of operations

$$\mathsf{Enc}(\mathsf{k}, \mathbb{C}^0, m_0), \ldots, \mathsf{Enc}(\mathsf{k}, \mathbb{C}^{n-1}, m_{n-1})$$

resulting in data structure $\mathbb{C}^n$ that

$$\forall i \, \exists j \; m_i = \mathsf{Dec}(\mathsf{k}, \mathbb{C}^n[j]).$$

[4] In case several cells of a simple data structure are encrypted as a whole, we call this combination a cell of another data structure.

[5] Note that in case of public key encryption our definition does not imply that the entire operation can be completed using only the public key.

For the correctness of search we expect that for any

$$\{j_0, \ldots, j_{\ell-1}\} := \mathsf{Search}(\mathsf{k}, \mathbb{C}, a, b)$$

it holds that

$$\forall j \in \{j_0, \ldots, j_{\ell-1}\} \Longrightarrow \mathsf{Dec}(\mathsf{k}, \mathbb{C}[j]) \in [a, b]$$

and

$$\forall j \in \{j | \mathsf{Dec}(\mathsf{k}, \mathbb{C}[j]) \in [a, b]\} \Longrightarrow j \in \{j_0, \ldots, j_{\ell-1}\}.$$

We can now finally define an efficiently searchable encrypted data structure.

**Definition 4** (ESEDS)**.**
*An efficiently searchable encrypted data structure* ESEDS *is a searchable encrypted data structure where the running time* $\tau$ *of* Search *is poly-logarithmic in* $n$ *(plus the size of the returned set of matching ciphertext indices) and the space* $\sigma$ *of* ESEDS *is linear in* $n$*:*

$$\tau(\mathsf{Search}) \leq \mathcal{O}(\mathrm{polylog}(n) + \ell)$$
$$\sigma(\mathsf{ESEDS}) = \mathcal{O}(n)$$

It is now clear that efficient search prevents encrypting the entire data structure and thereby achieving semantic (IND-CPA) security. Next, we give our definition of security that implies that each cell's data is encrypted with a semantically secure encryption scheme. Our security definition also prevents all plaintext guessing attacks of the type of Naveed et al. and Grubbs et al. Furthermore, we show that even when the data structure consists of only one semantically secure ciphertext in each cell, this does not guarantee security against these plaintext guessing attacks.

## 3 Security of ESEDS

Before we define the security of an ESEDS we will review recent attacks on cloud infrastructures and searchable encryption scheme to motivate our security model. Particular we review in depth plaintext guessing attacks that only need a (multi-)set of ciphertexts as input (and do not perform active attacks during encryption or search operations). We try to generalize these attacks and show that even if all elements in an ESEDS are semantically secure encrypted, this does not imply that these attacks are infeasible.

### 3.1 Motivation

Our model is motivated by recent attacks on cloud infrastructures and order-preserving or deterministic encryption. Not only the theoretic demonstrations, but also real world incidents show the risks of deterministic – not even order-preserving – encryption. In at least one case passwords were encrypted using

a deterministic algorithm and many subsequently broken [11]. The cryptanalysis was performed on stolen ciphertexts only (using additional plaintext hints). Many other hacking incidents have been recently publicized, e.g. [13, 26], that resulted in leakage of sensitive information – not necessarily ciphertexts.

All these attacks share a common "anatomy". The hackers are capable to break in, access and copy sensitive information. They used the opportunity of access to gain as much data as possible in a short time, i.e. the adversary obtains a *static* snapshot. Note that this does not rule out the revelation of more sophisticated, longitudinal attacks in the future, but underpins the pressure to secure our current systems.

In this respect our model achieves the following: An attacker gaining access to all ciphertexts stored in an encrypted database does not gain additional information without making any assumption about his background knowledge. We can even assume perfect background knowledge, i.e. the adversary has chosen all plaintexts. This may sound contradictory at first – why would someone break into a database which he entirely created. However, if we are able to show security against such strong adversaries, security holds even if the adversary has less, e.g. imperfect, background knowledge.

### 3.2 Security Definition

We give our security definition as an adaptation of semantic security to data structures. We show that our adaptation implies that each data value is semantically secure encrypted. However, we also show that even if all cells consist of only one semantically secure ciphertext, our adaptation is not necessarily fulfilled.

First, recall the definition of semantic security.

**Definition 5 (IND-CPA Security).** *A public-key encryption scheme* $\mathsf{PPKE}$ *has* indistinguishable encryptions under a chosen-plaintext attack*, or is* IND-CPA-secure*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{PPKE}}(\lambda) &:= \left|\Pr\left[\mathrm{Exp}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{PPKE}}(\lambda) = 1\right] - \frac{1}{2}\right| \\ &\leq \mathsf{negl}(\lambda)\end{aligned}$$

$\underline{\mathrm{Exp}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{PPKE}}(\lambda)}$
$\langle \mathsf{pk}, \mathsf{sk}\rangle \leftarrow \mathsf{PPKE.KGen}(1^\lambda)$
$\langle m_0, m_1, \mathsf{st}\rangle \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk})$
$b \leftarrow_\$ \{0,1\}$
$c \leftarrow \mathsf{PPKE.Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}, c, \mathsf{st})$
**return** $b = b'$

We note that IND-CPA-security only considers a single ciphertext whereas a data structure consists of multiple ciphertexts and hence some structural information. Exactly this structural information can be used in plaintext guessing attacks and we need to adapt semantic security to all ciphertexts. We call our adaptation *indistinguishability under chosen-plaintext attacks for data structures* or IND-CPA-DS-security for short. Loosely speaking, our security model ensures that an adversary who has chosen all plaintexts encrypted in a data structure cannot guess the plaintext of *any* ciphertext better than a random guess. We denote the size of multi-set $\mathbb{M}$ as $|\mathbb{M}|$ and the number of occurrences of element $m$ in multi-set $\mathbb{M}$ as $\#_{\mathbb{M}} m$.

**Definition 6 (IND-CPA-DS Security).**
*An efficiently searchable encrypted data structure* $\mathsf{ESEDS}$ *is* indistinguishable under a chosen-plaintext attack*, or is* IND-CPA-DS-secure*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{ESEDS}}(\lambda) &:= \left|\Pr\left[\mathrm{Exp}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{ESEDS}}(\lambda) = \langle 1, p\rangle\right] - p\right| \\ &\leq \mathsf{negl}(\lambda)\end{aligned}$$

$\mathrm{Exp}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{ESEDS}}(\lambda)$

$\langle \mathsf{pk}, \mathsf{sk}\rangle \leftarrow \mathsf{ESEDS.KGen}(1^\lambda)$
$\langle \mathbb{M}_0, \mathbb{M}_1, \mathsf{st}\rangle \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk})$
**if** $|\mathbb{M}_0| \neq |\mathbb{M}_1|$ **then return** $\perp$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathbb{C} := \varepsilon$
**foreach** $m \in \mathbb{M}_b$ **do**
  $\mathbb{C} \leftarrow \mathsf{ESEDS.Enc}(\mathsf{sk}, m, \mathbb{C})$
**endforeach**
$\langle j', m'\rangle \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}, \mathbb{C}, \mathsf{st})$
**return** $\left\langle \mathsf{ESEDS.Dec}(\mathsf{sk}, \mathbb{C}[j']) = m', \frac{\#_{\mathbb{M}_0 \cup \mathbb{M}_1} m'}{|\mathbb{M}_0 \cup \mathbb{M}_1|} \right\rangle$

There are two differences between IND-CPA-security and IND-CPA-DS-security. First, the adversary chooses two multi-sets of plaintexts as input to the challenge instead of two single plaintexts. This enables the adversary to create different situations to distinguish. Assume the adversary returns two disjoint multi-sets as $\mathbb{M}_0$ and $\mathbb{M}_1$, e.g. $\mathbb{M}_0 = \{0, 0\}$ and $\mathbb{M}_1 = \{1, 1\}$. Then it can attempt to distinguish which of the two plaintext multi-sets have been encrypted by guessing any plaintext in the data structure. Assume the adversary returns the same multi-set as $\mathbb{M}_0$ and $\mathbb{M}_1$, but with distinct plaintexts in the (identical) multi-set, e.g. $\mathbb{M}_0 = \mathbb{M}_1 = \{0, 1\}$. This is admissible in the definition of IND-CPA-DS-security, since the only requirement is that the two multi-sets are of the same size. The

adversary can then attempt to distinguish at which position in the data structure each plaintext has been encrypted.

In order to enable the adversary to win the game when the position in the data structure is not indistinguishable, we made a second change to IND-CPA-security: The adversary's guess is the plaintext of a single ciphertext at any position in the data structure. Hence, the adversary does not necessarily have to distinguish between the two plaintext multi-sets, it is sufficient, if it guesses correctly within the choice of sets (which may be equal). However, even if the position in the ciphertext is indistinguishable, in order to win the adversary only has to guess correctly with a probability non-negligibly better than the frequency of the plaintext in the *union of the multi-sets*. Hence, if the two multi-sets are not equal and the adversary can guess the chosen multi-set, it can win the game.

We next explain the implications of IND-CPA-DS-security and first prove that IND-CPA-DS-security implies IND-CPA-security. Our proof assumes the use of public-key encryption, but the proof for symmetric encryption is analogous using an encryption oracle. We prove this by turning an adversary $\mathcal{B}$ that has advantage $\epsilon$ in experiment $\mathrm{Exp}^{\text{IND-CPA}}_{\mathcal{B},\mathsf{PPKE}}$ into an adversary $\mathcal{A}$ that has advantage $\epsilon$ in experiment $\mathrm{Exp}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{ESEDS}_{\mathsf{PPKE}}}$.

**Theorem 1.** *If* $\mathsf{ESEDS}_{\mathsf{PPKE}}$ *is* IND-CPA-DS*-secure, then each ciphertext of the data element in* $\mathbb{C}[j]$ $(0 \leq j < n)$ *must be from a* IND-CPA*-secure encryption scheme.*

*Proof.* Let $\mathcal{B}$ be an adversary that has advantage $\epsilon$ in experiment $\mathrm{Exp}^{\text{IND-CPA}}_{\mathcal{B},\mathsf{PPKE}}$. We construct an adversary $\mathcal{A}$ for experiment $\mathrm{Exp}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{ESEDS}_{\mathsf{PPKE}}}$ as follows.

| $\mathcal{A}(1^\lambda, \mathsf{pk})$ | $\mathcal{A}(1^\lambda, \mathsf{pk}, \mathbb{C}, \mathsf{st})$ |
|---|---|
| $\langle m_0, m_1, \mathsf{st}' \rangle \leftarrow \mathcal{B}(1^\lambda, \mathsf{pk})$ | $\mathsf{st}' \Vert \{\mathbb{M}_0, \mathbb{M}_1\} := \mathsf{st}$ |
| $\mathbb{M}_0 := \{m_0\}$ | $b' \leftarrow \mathcal{B}(1^\lambda, \mathsf{pk}, \mathbb{C}[0], \mathsf{st}')$ |
| $\mathbb{M}_1 := \{m_1\}$ | **return** $\langle 0, \mathbb{M}_{b'}[0] \rangle$ |
| $\mathsf{st} := \mathsf{st}' \Vert \{\mathbb{M}_0, \mathbb{M}_1\}$ | |
| **return** $\langle \mathbb{M}_0, \mathbb{M}_1, \mathsf{st} \rangle$ | |

The adversary $\mathcal{B}$'s view is indistinguishable from experiment $\mathrm{Exp}^{\text{IND-CPA}}_{\mathcal{B},\mathsf{PPKE}}$ If adversary $\mathcal{B}$ guesses correctly, then $\mathcal{A}$'s output is also correct. Hence, if $\mathcal{B}$'s advantage is $\epsilon$, then $\mathcal{A}$'s advantage is $\epsilon$.

However, we also prove that even if each cell in $\mathbb{C}$ consists of a single ciphertext from a IND-CPA-secure, public-key encryption scheme $\mathsf{PPKE}$, then this does not imply IND-CPA-DS-security. We prove by giving a data structure that consists of a single ciphertext from $\mathsf{PPKE}$, but that is not IND-CPA-DS-secure. Again, the proof for symmetric encryption is analogous.

**Theorem 2.** *If each ciphertext in an efficiently searchable, encrypted data structure* $\mathsf{ESEDS}_{\mathsf{PPKE}}$ $\mathbb{C}[j]$ $(0 \leq j < n)$ *is from a* IND-CPA*-secure encryption scheme* $\mathsf{PPKE}$*, then* $\mathsf{ESEDS}_{\mathsf{PPKE}}$ *is not necessarily* IND-CPA-DS*-secure.*

*Proof.* Given a multi-set of plaintexts $\mathbb{M}$ and a IND-CPA-secure, public-key encryption scheme $\mathsf{PPKE}$, we construct a data structure as follows. Let rand-order($\mathrm{m_i}$) be the randomized order of each plaintext $m_i \in \mathbb{M}$. Recall that in a randomized order of a multi-set, elements are sorted, but ties are broken based on the outcome of a coin flip.

$$\mathbb{C}[\text{rand-order}(m_i)] \leftarrow \mathsf{PPKE.Enc}(\mathsf{pk}, m_i)$$

This data structure has equivalent leakage to frequency-hiding order-preserving encryption (FH-OPE) by Kerschbaum [21]. It is easy to see that each cell of the data structure consists of only one semantically secure ciphertext. However, we construct an adversary that succeeds with probability 1 for $p = \frac{1}{2}$ in our experiment $\mathrm{Exp}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{ESEDS}_{\mathsf{PPKE}}}$.

| $\mathcal{A}(1^\lambda, \mathsf{pk})$ | $\mathcal{A}(1^\lambda, \mathsf{pk}, \mathbb{C}, \mathsf{st})$ |
|---|---|
| $\mathbb{M} := \{0, 1\}$ | **return** $\langle 0, 0 \rangle$ |
| **return** $\langle \mathbb{M}, \mathbb{M}, \varepsilon \rangle$ | |

The adversary always wins the game, since in the given encryption scheme plaintext 0 will always be encrypted at position 0. Grubbs et al. showed in [16] the practicality of the attack by constructing a plaintext guessing attack on FH-OPE. In their experiments it succeeds with probability 30% where the base line guessing probability is only 4%.

**Relation to Other Security Definitions** In searchable encryption a security definition of indistinguishability under chosen-keyword attack (IND-CKA-security) has been defined in [9] and used in many subsequent works. Loosely speaking, this security definition states that the data structure is IND-CKA-secure, if it is indistinguishable from a simulator given (a set of) leakage function(s) $\mathcal{L}$. However, this can be misleading, since the leakage function does not necessarily clearly state the impact on plaintext guessing attacks. We first state the following corollary:

**Corollary 1.** *If a public-key encryption scheme* $\mathsf{PPKE}$ *is* IND-CPA-secure*, then there exists a simulator* $\mathsf{Sim}_{\mathsf{PPKE}}(1^\lambda, \mathsf{pk})$*, such that for all PPT adversaries* $\mathcal{A}$ *and all PPT distinguishers* $\mathsf{Dist}$

$$\begin{aligned}\mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{Dist},\mathsf{PPKE}}(\lambda) := \\ \Big|\Pr\Big[\mathsf{Dist}(c, \mathsf{pk}) = 1 : \langle c, \mathsf{pk}\rangle \leftarrow \mathrm{RealExp}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{PPKE}}(\lambda)\Big] - \\ \Pr\big[\mathsf{Dist}(c, \mathsf{pk}) = 1 : \langle c, \mathsf{pk}\rangle \leftarrow \mathrm{SimExp}^{\text{IND-CPA}}_{\mathsf{Sim}_{\mathsf{PPKE}},\mathsf{PPKE}}(\lambda)\big]\Big| \\ \leq \mathsf{negl}(\lambda)\end{aligned}$$

$\underline{\mathrm{RealExp}^{\text{IND-CPA}}_{\mathcal{A},\mathsf{PPKE}}(\lambda)}$

$\langle \mathsf{pk}, \mathsf{sk} \rangle \leftarrow \mathsf{PPKE.KGen}(1^\lambda)$
$\langle m_0, m_1, \mathsf{st} \rangle \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk})$
$b \leftarrow\!\!\$ \{0,1\}$
$c \leftarrow \mathsf{PPKE.Enc}(\mathsf{pk}, m_b)$
**return** $c, \mathsf{pk}$

$\underline{\mathrm{SimExp}^{\text{IND-CPA}}_{\mathsf{Sim}_{\mathsf{PPKE}},\mathsf{PPKE}}(\lambda)}$

$\langle \mathsf{pk}, \mathsf{sk} \rangle \leftarrow \mathsf{PPKE.KGen}(1^\lambda)$
$c \leftarrow \mathsf{Sim}_{\mathsf{PPKE}}(1^\lambda, \mathsf{pk})$
**return** $c, \mathsf{pk}$

It follows that there exists a simulator for an encrypted data structure whose cells consists only of semantically secure ciphertexts which requires a leakage function of only the length $n$ of the data structure and the public key $\mathsf{pk}$. However, as we have shown in Theorem 2 such a data structure may not be IND-CPA-DS-secure and susceptible to plaintext guessing attacks.

**Theorem 3.** *An efficiently searchable, encrypted data structure* $\mathsf{ESEDS}_{\mathsf{PPKE}}$ *may be indistinguishably simulated with a leakage function* $\mathcal{L} = \{\mathsf{pk}, n\}$ *and be susceptible to plaintext guessing attacks.*

*Proof.* Consider the data structure from the proof of Theorem 2. It is indistinguishable from $n = |\mathbb{M}|$ ciphertexts produced using public key $\mathsf{pk}$ and successful plaintext guessing attacks have been shown by Grubbs et al. in [16].

Hence, leaking the number of plaintexts may be sufficient for a successful plaintext guessing attack in a simulation-based security proof. Our IND-CPA-DS-security model prevents this by introducing a *structural independence* constraint. While Curtmola et al. have been careful not to make this mistake in [9] and their $\mathsf{ESEDS}$ is IND-CPA-DS-secure, subsequent work was not as careful. Boelter et al.'s data structure [2] has a (correct) simulation-based proof and is not IND-CPA-DS-secure and susceptible to plaintext guessing attacks.[6]

**Impact on Plaintext Guessing Attacks** We can now revisit the plaintext guessing attacks on deterministic and order-preserving encryption. First, our security model fully captures the attack setup. The adversary is given full ciphertext information and can chose the plaintexts such that it has perfect background knowledge[7], i.e. the adversary in our model has at least the same information as was used in those attacks. Second, our security definition implies that if the adversary is then able to infer even one plaintext better than with negligible probability over guessing our scheme is broken. Hence security in the IND-CPA-DS model implies security against all (passive, ciphertext-only) plaintext guessing attacks.

[6] This is easy to see, since they do not encrypt the structural information in their data structure, i.e. the pointers to leaf nodes in the tree, and hence the ciphertexts can be ordered.

[7] Recall that the adversary is allowed to submit the same plaintext multi-sets in the IND-CPA-DS-security experiment

## 4 An IND-CPA-DS-Secure ESEDS for Range Queries

We next present our efficiently searchable, encrypted data structure for range queries that is IND-CPA-DS-secure. We emphasize that using the result from the data structure we can perform range queries in any commodity database management system without modifications. Hence, our data structure is as easy to integrate as order-preserving encryption, yet it is secure against chosen-plaintext attacks. We begin by describing the system architecture and give the intuition of our construction.

In our setup we assume a client holding the secret key $\mathsf{k} \leftarrow \mathsf{ESEDS.KGen}(1^\lambda)$ and a server that holds the data structure $\mathbb{C}$. The server may hold several data structures managed independently for each database column, but needs to take care of correlation attacks as in [12]. A database table then contains the rows linking the entries by their index in the data structure. After encrypting the plaintexts, the client and the server can interactively perform a search query, e.g. a range query, on the server's data structure which results in two indices $j, j'$. Then these two indices $j, j'$ can be used in subsequent range queries on the database management system. We assume that the server is *semi-honest*, i.e. only performs *passive* attacks. This model is commonly assumed in the scientific literature on database security.

### 4.1 Intuition

Our data structure combines the ideas of three previous order-preserving encryption schemes: First, the scheme by Popa et al. [28] provides the basis for managing the order of ciphertexts in a stateful, interactive manner. Of course, this scheme is not secure against the attacks by Naveed et al., since it is deterministic and ordered. Second, we add the frequency-hiding aspect of the scheme by Kerschbaum [21]. The scheme itself cannot be used as the basis of an IND-CPA-DS-secure data structure, since it partially leaks the insertion order. Therefore the frequency-hiding idea needs to be fit into Popa et al.'s scheme. We do this by encrypting the plaintext using a probabilistic algorithm (similar to the stOPE scheme in [28]) and also inserting a ciphertext for each plaintext using Kerschbaum's random tree traversal. This combined construction would still not be IND-CPA-DS secure. Third, we apply Boldyreva et al.'s modular order-preserving encryption idea [4]. This idea rotates the plaintexts around a modulus statically hiding the order. However, modular order-preserving encryption has been developed for *deterministic* order-preserving encryption. In our probabilistic encryption – as introduced by Kerschbaum – we need to apply the modulus on the ciphertexts. This can be done by updating the modulus after encryption.

In summary, intuitively our encryption scheme works as follows: We maintain a list of ciphertexts for each plaintext (including duplicates) sorted by the plaintexts on the server. However, the list is rotated around a random offset (chosen uniformly from the range between 1 and the number of ciphertexts). We then encrypt and search using binary search. However, due to the rotation which can

divide a set of identical plaintexts adjacent in the list into a lower and upper part, the search and encryption algorithms become significantly more complex which is apparent in their detailed description below.

## 5 Encryption Algorithm

Let $\mathsf{PE}$ be a standard, probabilistic encryption scheme supporting the following three – possibly probabilistic – polynomial-time algorithms: $\mathsf{KGen}$, $\mathsf{Enc}$ and $\mathsf{Dec}$. We use symmetric encryption, e.g. AES in CBC or GCM mode, for speed, but assume an encryption oracle in the definition of semantic security. Let $\mathbb{D}$ be the domain of plaintexts and $N = |\mathbb{D}|$ its size. We now describe the algorithms and protocols of our efficiently searchable, encrypted data structure:

- $\mathsf{k} \leftarrow \mathsf{KGen}(1^\lambda)$: Execute $\mathsf{k} \leftarrow \mathsf{PSE.KGen}(1^\lambda)$.
- $\mathbb{C}^{h+1} \leftarrow \mathsf{Enc}(\mathsf{k}, m, \mathbb{C}^h)$: We denote $\mathbb{C}^h$ as $\mathbb{C}$ for brevity, if it is clear from the context. First the client and server identify the index $j_m$ where $m$ is to be inserted (before). Then the client sends the ciphertext of $m$ to the server which inserts it at position $j_m$. Finally the server rotates the data structure by a random offset.
  1. The client sets $l := 0$ and $u := n - 1$.
  2. If $n = 0$ then go to step 5.
  3. The client requests $\mathbb{C}[0]$ and sets $r := \mathsf{Dec}(\mathsf{k}, \mathbb{C}[0])$.
  4. Set $j := \lfloor l + \frac{u-l}{2} \rfloor$. The client requests $\mathbb{C}[j]$ and executes $m' := \mathsf{Dec}(\mathsf{k}, \mathbb{C}[j])$. If $m' - r \bmod N > m - r \bmod N$, then the client sets $l := j + 1$. If $m' - r \bmod N < m - r \bmod N$, then the client sets $u := j$. If $m' = m \bmod N$, then the client flips a random coin and sets either $l := j + 1$ or $u := j$ depending on the outcome of the coin flip. The client repeats this step until $l = u$.
  5. The client sends $c \leftarrow \mathsf{PSE.Enc}(\mathsf{k}, m)$ to the server.
  6. The server sets

$$\begin{aligned} \mathbb{C}[n] &:= \mathbb{C}[n-1] \\ \mathbb{C}[n-1] &:= \mathbb{C}[n-2] \\ &\cdots \\ \mathbb{C}[l+1] &:= \mathbb{C}[l] \\ \mathbb{C}[l] &:= c \end{aligned}$$

     The server sets $n := n + 1$.
  7. The server chooses a random number $s \leftarrow_\$ \mathbb{Z}_{n-1}$. The server sets the new encrypted data structure to $\mathbb{C}^{h+1}[j] := \mathbb{C}[j + s \bmod n]$ for $0 \le j < n$ as a result of the encryption operation. This data structure $\mathbb{C}^{h+1}$ will be used as input to the next encryption operation.
- $\langle j, j' \rangle := \mathsf{Search}(\mathsf{k}, \mathbb{C}, a, b)$: Wlog. we assume that $a \le b$ in the further exposition. In case $a > b$ the query is rewritten as to match all $x$, such that $0 \le x < b \vee a \le x < N$.

Let $j_{\min}(v)$ be the minimal index of plaintext $v$ and $j_{\max}(v)$ be the maximal index of plaintext $v$.

$$j_{\min}(v) := \min(j|\mathsf{Dec}(\mathsf{k}, \mathbb{C}[j]) = v)$$

$$j_{\max}(v) := \max(j|\mathsf{Dec}(\mathsf{k}, \mathbb{C}[j]) = v)$$

If $j_{\min}(v) = 0$ and $j_{\max}(v) = n - 1$ and there are two distinct plaintexts in the data structure, then we redefine as

$$j_{\min}(v) := j + 1|\mathsf{Dec}(\mathsf{k}, \mathbb{C}[j]) < v \wedge \mathsf{Dec}(\mathsf{k}, \mathbb{C}[j + 1]) = v)$$

$$j_{\max}(v) := j - 1|\mathsf{Dec}(\mathsf{k}, \mathbb{C}[j]) > v \wedge \mathsf{Dec}(\mathsf{k}, \mathbb{C}[j - 1]) = v)$$

If $a$ and $b$ do not span the modulus, i.e. $j_{\min}(a) < j_{\max}(b)$, then a query for $x \in [a, b]$ is rewritten to $j_{\min}(a) \leq x \leq j_{\max}(b)$. Else, it is rewritten to $0 \leq x < j_{\max}(b) \vee j_{\min}(a) \leq x < n'$.
Both $j_{\min}(a)$ and $j_{\max}(b)$ are found using a separately run, interactive binary search. We next present this protocol.

1. The client sets $l := 0$ and $u := n - 1$.
2. The client requests $\mathbb{C}[0], \mathbb{C}[n-1]$ and sets $r := \mathsf{Dec}(\mathsf{k}, \mathbb{C}[0])$. If $\mathsf{Dec}(\mathsf{k}, \mathbb{C}[0]) = \mathsf{Dec}(\mathsf{k}, \mathbb{C}[n - 1])$ and searching for $j_{\min}(a)$, it sets $r := r + 1$.
3. Set $j := \lfloor l + \frac{u-l}{2} \rfloor$. The client requests $\mathbb{C}[j]$ and executes $m := \mathsf{Dec}(\mathsf{k}, \mathbb{C}[j])$. If $m - r \bmod N < a - r \bmod N$ (or $m - r \bmod N \leq b - r \bmod N$, respectively) then the client sets $l := j + 1$. Else the client sets $u := j$. The client repeats this step until $l = u$.
4. The client returns $j_{\min}(a) := l$ (or $j_{\max}(b) := u$, respectively).

– $m := \mathsf{Dec}(\mathsf{k}, \mathbb{C}[j])$: Set $m := \mathsf{PSE.Dec}(\mathsf{k}, \mathbb{C}[j])$.

### 5.1 Security

**Theorem 4.** *Our efficiently searchable, encrypted data structure* $\mathsf{ESEDS}_{\mathsf{PSE}}$ *is* IND-CPA-DS*-secure.*

*Proof.* Since all cells of the data structure consists only of ciphertexts from a IND-CPA-secure encryption scheme, we can replace the encrypted data structure by a simulator. Let the simulator $\mathsf{Sim}^{\mathrm{E_k}}_{\mathsf{ESEDS}}(1^\lambda, n)$ output $n$ ciphertexts $c \leftarrow \mathrm{E_k}(0)$. The adversary $\mathcal{A}$ cannot distinguish the following experiment $\mathrm{Exp}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{Sim}_{\mathsf{ESEDS}},\mathsf{ESEDS}_{\mathsf{PSE}}}$ from experiment $\mathrm{Exp}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{ESEDS}_{\mathsf{PSE}}}$ except with negligible probability.

$\underline{\mathrm{Exp}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{Sim}_{\mathsf{ESEDS}},\mathsf{ESEDS}_{\mathsf{PSE}}}(\lambda)}$

$\langle \mathsf{k}, \mathbb{C} \rangle \leftarrow \mathsf{ESEDS.KGen}(1^\lambda)$

$\langle \mathbb{M}_0, \mathbb{M}_1, \mathsf{st} \rangle \leftarrow \mathcal{A}^{\mathrm{E_k}}(1^\lambda)$

**if** $|\mathbb{M}_0| \neq |\mathbb{M}_1|$ **then return** $\perp$

$\mathbb{C} \leftarrow \mathsf{Sim}^{\mathrm{E_k}}_{\mathsf{ESEDS}}(1^\lambda, |\mathbb{M}_0|)$

$\langle j', m' \rangle \leftarrow \mathcal{A}^{\mathrm{E_k}}(1^\lambda, \mathbb{C}, \mathsf{st})$

**return** $\left\langle \mathsf{ESEDS.Dec}(\mathbb{C}[j']) = m', \frac{\#_{\mathbb{M}_0 \cup \mathbb{M}_1} m'}{|\mathbb{M}_0 \cup \mathbb{M}_1|} \right\rangle$

$\underline{\mathrm{E_k}(m)}$

$c \leftarrow \mathsf{PSE.Enc}(\mathsf{k}, m)$

**return** $c$

The adversary $\mathcal{A}$ in $\mathrm{Exp}^{\text{IND-CPA-DS}}_{\mathcal{A},\mathsf{Sim}_{\mathsf{ESEDS}},\mathsf{ESEDS}_{\mathsf{PSE}}}$ clearly has no information which plaintext multi-set has been encrypted or about the plaintexts' positions in the data structure. Since in our $\mathsf{ESEDS}_{\mathsf{PSE}}$ each plaintext has equal probability of being at any index within the data structure, the adversary can at best guess the index $j'$ for any $m' \in \mathbb{M}$. However, the probability of a successful guess is bounded by $\frac{\#_{\mathbb{M}_0 \cup \mathbb{M}_1} m'}{|\mathbb{M}_0 \cup \mathbb{M}_1|}$.

## 6 Performance Evaluation

We prototypically implemented and in a number of experiments evaluated the performance our IND-CPA-DS-secure $\mathsf{ESEDS}$. In this section we report the results of our experiments measuring the run-time of range searching over encrypted data.

### 6.1 Implementation

We used Java for our implementation and evaluation, since many multi-tier applications are implemented in Java. Although a native cryptographic library, such as Intel's AES-NI, promises further performance improvements, programming languages such as C or C++ are more commonly used for systems software (such as database management systems) rather than for database applications (which only issue database queries). However, in our setup encryption and decryption is performed in the database application. We used Oracle's Java 1.8 and all experiments were run on the Java SE 64-Bit Server virtual machine. The database backend was the MySQL replacement MariaDB in version 10.1. When using a database, such as MariaDB, that was not specifically developed for operation on encrypted data, one needs to configure it to prevent the attacks on configuration described by Grubbs et al. [15]. All experiments were run on a single machine with a 4-core Intel i7 CPU at 2.9 GHz and 16 GB of RAM on Windows 10 Enterprise.

### 6.2 Experimental Setup

We measure the run-time of a typical, simply structured (i.e. a single search term and no conjunctions or disjunctions) database query on a single ordered database column, e.g. a range query or a top-k query. We use synthetic data and queries. However, we adapt our choice of parameters to the data from the DBLP data set [1]. In the spirit of Grubbs et al. [16] we considered author names. At the time of our experiments there were about 1.500.000 million distinct author names in DBLP, the most frequent of which appears roughly 80 times.

We implement the client interface as it would be used in an application using a database. The application supplies the parameters, e.g. the start $a$ and end $b$ of a range or the $k$ in top-k, and receives the results in plaintext. Thus, our measured run-time includes the $\mathsf{Search}$ algorithm, the standard query by the

database management system and the decryption of the result. We emphasize that in more complex queries, e.g. including multiple search terms combined by conjunction and disjunctions, the relative time for executing the query on the database management system would be proportionally higher. Hence, our experiments put an upper bound on the worst case of the relative overhead, since our queries are of the simplest form.

Our target quantity in our measurements is the absolute run-time in milliseconds. For range queries we measure the dependence of the run-time on different parameters.

- *Size of the database*: We vary the database size from 100.000 to 1.000.000 plaintexts in steps of 100.000, i.e. data items before encryption.
- *Size of the queried range*: We vary the range size and consequently the result set size in the query from 10 to 100 in steps of 10.

For top-$k$ queries we measure the dependence of the run-time of the following parameter.

- $k$: We vary the limit $k$ from 10 to 100 in steps of 10.

We compare the run-time on encrypted data to the run-time on plaintext data. Note that queries on plaintext only need to execute the query on the database management system, i.e. the time for the Search algorithm and decryption of results is 0.

We use synthetically generated data and queries. We uniformly choose distinct plaintexts and we uniformly choose a begin of the range query and then compute the end using the fixed size parameter of the experiment.

We repeat each experiment 30 times discarding the first 10 experiments in order to allow to adjust the Java JIT compiler. We report the mean and 95% confidence interval for each parameter setting.

### 6.3 Results

*Database size*: Figure 1 shows the running time over the database size. We use a query range size of 10. The database size increases from 100.000 to 1.000.000 plaintexts in steps of 100.000. The running time is measured in milliseconds. The error bars show the 95% confidence interval. Since our search algorithms run in sub-linear time only a very slight increase (20%) in running time is measurable compared to the increase in database size (900%). The overhead of our encryption is roughly 9 milliseconds.

*Query Range Size*: Figure 1 shows the running time over the query range size. We use a database size of 1.000.000 plaintexts. The query range size and hence the expected result set size increases from 10 to 100 in steps of 10. The running time is measured in milliseconds. The error bars show the 95% confidence interval. The running time increase is slight and approximately linear in the query range size and there is a constant baseline. We attribute the constant cost to our binary search algorithm which as shown in Figure 1 behaves almost constant for

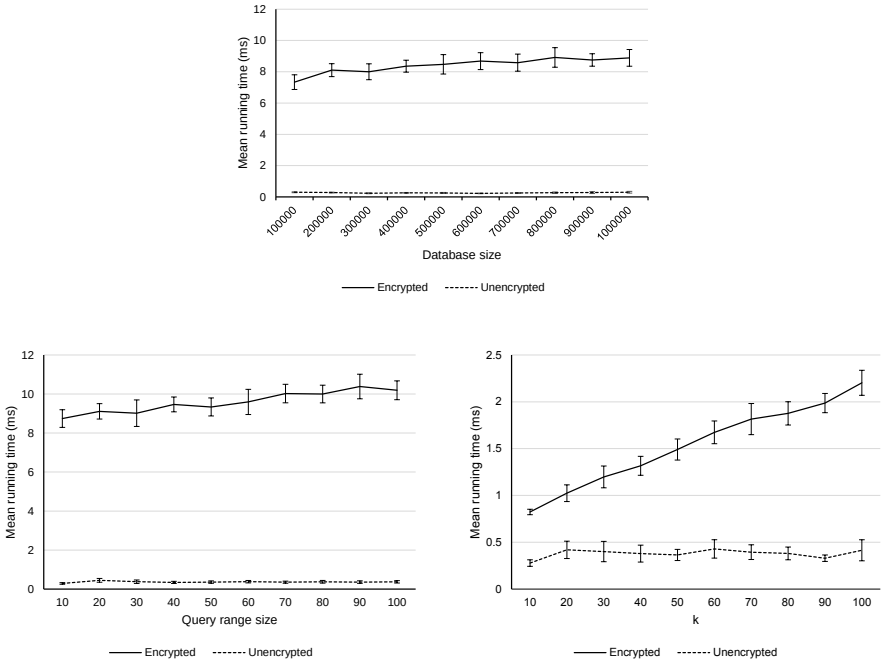


**Fig. 1.** (top) Performance over different database sizes (left) Performance over query range size (right) Performance over k for top-k queries

these database sizes. We attribute this increase to the cost of decryption which is dominated by the cryptographic operations.

*Top-k queries*: Figure 1 shows the running time for top-$k$ queries over $k$. We use a database size of 1.000.000 plaintexts. The value of $k$ increases from 10 to 100 in steps of 10. The running time is measured in milliseconds. The error bars show the 95% confidence interval. The constant baseline is lower, since top-$k$ queries can be executed without a search algorithm, when the minimum ciphertext (the rotation value) is stored as part of the key. The linear increase due to decryption of results is now clearly visible.

## 7 Related Work

Our work is related to other searchable encryption schemes – particularly for range queries –, order-revealing encryption and leakage-abuse attacks.

### 7.1 Searchable Encryption

Searchable encryption allows the comparison of a token (corresponding to a plaintext) to a ciphertext. The ciphertext (without any token) is IND-CPA-

secure. The token can match plaintexts for equality or the plaintext to a range. Only the secret key holder can create tokens.

Tackling range queries with searchable encryption is complex. The first proposal by Boneh and Waters in [6] had ciphertext size linear in the size of the domain of the plaintext. The first poly-logarithmic sized ciphertexts scheme was proposed by Shi et al. in [31]. However, their security model is somewhat weaker than standard searchable encryption. The construction is based on inner-product predicate encryption which has been made fully secure by Katz et al. in [19]. All schemes follow the construction by Song et al. (without inverted indices) and require linear search time. The first attempt to build range-searchable encryption into an index (an ESEDS) has been made by Lu in [25]. However, the inverted index tree reveals the pointers and is hence no more secure than order-preserving encryption. Demertzis et al. [10] map a range query to keyword queries by providing tradeoffs between storing replicated values in each of its ranges and enumerating all values within range query. The search can then be easily performed using the data structure of Curtmola et al. [9]. While the scheme is range searchable, its queries are very revealing and it has high storage cost (at least $O(n \log n)$). Boelter et al. [2] use garbled circuits to implement the search within a node of the index. They do not encrypt the pointers in the index and are hence susceptible to the attacks by Naveed et al. and are not IND-CPA-DS secure. The scheme by Hahn and Kerschbaum [17] creates an index using the access pattern of the range queries. No other information is leaked, however, this provides amortized poly-logarithmic search time. The scheme is only IND-CPA-DS-secure as long as no queries have been performed (and the index has been partially built). Their scheme is based on inner-product predicate encryption which is too slow for practical use.

### 7.2 Order-Revealing Encryption

Order-revealing encryption [5, 8, 24] is an alternative to order-preserving encryption. Instead of preserving the order there is a public function that reveals the order of two plaintexts using the ciphertexts only. At first, it may seem paradoxical to combine the disadvantages of order-preserving and searchable encryption: order revelation and modified comparison function. However, order-revealing encryption has also advantages. It allows an IND-OCPA secure (as defined in [3]) encryption with constant-size ciphertexts, constant size client storage and without mutation circumventing impossibility results in [3] and [28]. However, the first construction was not only impractical due to its disadvantages, but also due to its performance. A different construction with slightly more leakage, but significantly better performance was presented by Chenette et al. in [8]. This construction was further improved by Lewi and Wu in [24]. They allow comparison only between a token and an IND-CPA-secure ciphertext as in searchable encryption, i.e. the scheme has no leakage when no token is revealed. Their search procedure requires a linear search over all ciphertext and no indexing is possible. Hence, compared to our scheme which has logarithmic search time, order-revealing encryption currently remains impractical.

### 7.3 Leakage-Abuse Attacks

We discussed many leakage-abuse attacks on search over encrypted data. There are static attacks on order-preserving encryption [12, 16, 27, 29] and attacks using dynamic information that also work on searchable encryption [7, 14, 18, 20, 23, 32].

Kellaris et al. [20] have presented generic inference attacks on encrypted data using range queries. Their attacks work in a setup where the adversary has compromised the database server and can observe all queries, i.e. they work for dynamic leakage during the execution of queries and are not ciphertext-only attacks. They do not assume a specific cryptographic protection mechanism, but work only on its dynamic leakage profile, such as the access pattern or the result size, i.e. they also apply to ORAM-protected databases. The prerequisite assumption for Kellaris et al.'s attack to work is that the distribution of queries and the distribution of plaintexts differ. Specifically, they assume that each possible query will be executed, but not each possible plaintext is in the database. We note that Kellaris et al. performed all their attacks on synthetic data and queries whereas static ciphertext-only attacks on real data have been publicized [11].

There are also some more specific inference attacks. Islam et al. [18] and Cash et al. [7] have performed inference attacks by observing the queries on encrypted data. Islam et al. assume that the distribution of query keywords is approximately known and then can recover the query keywords using frequency analysis. Cash et al. improve the accuracy of this attack even under slightly weaker assumptions about the knowledge of query distribution, but then also use the information to recover plaintexts from the access pattern. Lacharite et al. [23] improve the accuracy of plaintext guessing by incorporating information from observed queries.

Next to the ones already discussed plaintext guessing attacks Pouliot and Wright show that adding deterministic encryption to Bloom filters – not surprisingly – does not prevent cryptanalysis [29]. Zhang et al. assume that the adversary can actively insert plaintexts and can then recover query plaintexts from the access pattern [32]. Grubbs et al. [14] also attacked an implementation of multi-user searchable encryption which allows inferences between users leading to a complete breakdown of the security guarantee of encrypted web applications.

Kolesnikov and Shikfa proposed to limit the querier's abilities to increase the security of order-preserving encryption [22]. In this paper we extend this idea and formally show that if we exclude queries, the static leakage from order-preserving ciphertexts (an ESEDS) can be negligible.

## 8 Conclusions

We give a new definition of security against plaintext guessing attacks by a snapshot adversary and show that it is insufficent to prove that all data items

have been probabilistically encrypted in order to be secure against these attacks. Then we prove by construction that the new definition can be fulfilled by an encrypted data structure, i.e. one secure against plaintext guessing attacks.

## References


1. http://dblp.l3s.de/dblp++.php
2. Boelter, T., Poddar, R., Popa, R.A.: A secure one-roundtrip index for range queries. Tech. Rep. 568, IACR Cryptology ePrint Archive (2016)
3. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Proceedings of the 28th International Conference on Advances in Cryptology. EUROCRYPT (2009)
4. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In: Proceedings of the 31st International Conference on Advances in Cryptology. CRYPTO (2011)
5. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Proceedings of the 34th International Conference on Advances in Cryptology. EUROCRYPT (2015)
6. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Proceedings of the 4th Theory of Cryptography Conference. TCC (2007)
7. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: Proceedings of the 22nd ACM Conference on Computer and Communications Security. CCS (2015)
8. Chenette, N., Lewi, K., Weis, S., Wu, D.: Practical order-revealing encryption with limited leakage. In: Proceedings of the 23rd International Workshop on Fast Software Encryption. FSE (2016)
9. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. Journal of Computer Security **19**(5) (2011)
10. Demertzis, I., Papadopoulos, S., Papapetrou, O., Deligiannakis, A., Garofalakis, M.: Practical private range search revisited. In: Proceedings of the ACM International Conference on Management of Data. SIGMOD (2016)
11. Ducklin, P.: Anatomy of a password disaster – adobe's giant-sized cryptographic blunder. https://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/ (2013)
12. Durak, B., DuBuisson, T., Cash, D.: What else is revealed by order-revealing encryption? In: Proceedings of the 23rd ACM Conference on Computer and Communications Security. CCS (2016)
13. Fitzpatrick, A.: Apple says systems weren't hacked in nude pics grab. http://time.com/3257945/apple-icloud-brute-force-jennifer-lawrence/ (2014)
14. Grubbs, P., McPherson, R., Naveed, M., Ristenpart, T., Shmatikov, V.: Breaking web applications built on top of encrypted data. In: Proceedings of the 23rd ACM Conference on Computer and Communications Security. CCS (2016)
15. Grubbs, P., Ristenpart, T., Shmatikov, V.: Why your encrypted database is not secure. Tech. Rep. 468, IACR Cryptology ePrint Archive (2017)
16. Grubbs, P., Sekniqi, K., Bindschaedler, V., Naveed, M., Ristenpart, T.: Leakage-abuse attacks against order-revealing encryption. Tech. Rep. 895, IACR Cryptology ePrint Archive (2016)

17. Hahn, F., Kerschbaum, F.: Poly-logarithmic range queries on encrypted data with small leakage. In: Proceedings of the ACM Workshop on Cloud Computing Security Workshop. CCSW (2016)
18. Islam, M., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In: Proceedings of the 19th Network and Distributed System Security Symposium. NDSS (2012)
19. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Advances in Cryptology. EUROCRYPT (2008)
20. Kellaris, G., Kollios, G., Nissim, K., O'Neill, A.: Generic attacks on secure outsourced databases. In: Proceedings of the 23rd ACM Conference on Computer and Communications Security. CCS (2016)
21. Kerschbaum, F.: Frequency-hiding order-preserving encryption. In: Proceedings of the 22nd ACM Conference on Computer and Communications Security. CCS (2015)
22. Kolesnikov, V., Shikfa, A.: On the limits of privacy provided by order-preserving encryption. Bell Labs Technical Journal **17**(3) (2012)
23. Lacharité, M.S., Minaud, B., Paterson, K.: Improved reconstruction attacks on encrypted data using range query leakage. Tech. Rep. 701, IACR Cryptology ePrint Archive (2017)
24. Lewi, K., Wu, D.: Order-revealing encryption: New constructions, applications, and lower bounds. In: Proceedings of the 23rd ACM Conference on Computer and Communications Security. CCS (2016)
25. Lu, Y.: Privacy-preserving logarithmic-time search on encrypted data in cloud. In: Proceedings of the 19th Network and Distributed System Security Symposium. NDSS (2012)
26. McCarthy, K.: Panama papers hack: unpatched wordpress, drupal bugs to blame? `http://www.theregister.co.uk/2016/04/07/panama_papers_unpatched_wordpress_drupal/` (2016)
27. Naveed, M., Kamara, S., Wright, C.V.: Inference attacks on property-preserving encrypted databases. In: Proceedings of the 22nd ACM Conference on Computer and Communications Security. CCS (2015)
28. Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. In: 34th IEEE Symposium on Security and Privacy. S&P (2013)
29. Pouliot, D., Wright, C.: The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In: Proceedings of the 23rd ACM Conference on Computer and Communications Security. CCS (2016)
30. Roche, D., Apon, D., Choi, S., Yerukhimovich, A.: Pope: Partial order preserving encoding. In: Proceedings of the 23rd ACM Conference on Computer and Communications Security. CCS (2016)
31. Shi, E., Bethencourt, J., Chan, H.T.H., Song, D.X., Perrig, A.: Multi-dimensional range query over encrypted data. In: Proceedings of the 2007 Symposium on Security and Privacy. S&P (2007)
32. Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: the power of file-injection attacks on searchable encryption. In: Proceedings of the 25th USENIX Security Symposium. USENIX SECURITY (2016)