Searchable Symmetric Encryption for Restricted Search

Máté Horváth, and István Vajda

Abstract—**The proliferation of cloud computing highlights the importance of techniques that permit both secure storage of sensitive data and flexible data management at the same time. One line of research with this double motivation is the study of Searchable Symmetric Encryption (SSE) that has provided several outstanding results in the recent years. These solutions achieve sublinear keyword search in huge databases by using various data structures to store keywords and document identifiers. In this work, we focus on certain scenarios in which search over the whole database is not necessary and show that the otherwise inefficient sequential scan (in linear time) can be very practical. This is due to the fact that adding new entries to the database comes for free in this case while updating a complex data structure without information leakage is rather complicated. To demonstrate the practicality of our approach we build a simple SSE scheme based on bilinear pairings and prove its security against adaptive chosen-keyword attacks in the standard model under the widely used Symmetric eXternal Diffie-Hellman (SXDH) assumption.**

Index Terms—**Searchable Symmetric Encryption, Forward Index, Type-3 Pairings, MAC.**

I. Introduction

COMPUTATION on hidden data and especially Searchable Symmetric Encryption (SSE) has become an extensively studied area of cryptography in the last more than one and a half decade, since the appearance of the seminal paper of [17]. The concept of SSE allows the secure storage of sensitive data on untrusted servers in the cloud without losing all the flexibility that plaintext data would allow. More precisely it supports keyword search over the ciphertexts in the following way: encrypted queries called trapdoors can be sent to the server which can test whether any of the stored ciphertexts matches the keyword underlying the trapdoor.

The two natural approaches towards realizing SSE are called “forward” and “inverted index”. The first one is to attach (or even include) one-way mappings of searchable keywords to the encrypted data. This leads to linear search complexity in the number of documents as the server has to go through all of them with a sequential scan to find all the matches for a trapdoor. A more sophisticated arrangement of the ciphertexts is to build an “inverted index”. In this case, the documents (or their IDs) are sorted based on the one-way mappings of keywords which are related to them. The latter solution allows logarithmic search complexity in the number of keywords. This clear benefit caused that the inverted index approach became prevalent in SSE design which made significant progress in the past years [19]. At the same time these solutions are rather complex and while operating smoothly on huge *static* databases (DB), handling the *rapid expansion* of the DB turns out to be more troublesome as the underlying data structure has to be updated without information leakage (see Table I for details).

In this work, we are looking for a simple solution to specific problems in the domain of SSE for which the inverted index based approach is not practical. Imagine the following scenarios!

Scenario 1. A device, deployed in an untrusted environment stores its own event logs in an SSE encrypted form with the event type as a keyword and possibly with a public time-stamp. It is often plausible to assume that the different events occur relatively often, resulting in frequent updates, and a remote operator is more likely to search for a specific event in a given time frame than in the entire DB. He can send a trapdoor together with a tract of time to the device that checks its encrypted DB and replies with the positive test results.

Scenario 2. The onboard unit of a vehicle regularly sends encrypted aggregate data to an honest but curious remote server. The data packets are tagged with a few predefined characteristic features of the given packet e.g., the oil level was under the limit, speeding was detected, the seatbelt was not fastened etc. An authority, possibly maintained by the car manufacturer as part of their services, can issue trapdoors for the car service/insurance company/police who can send these to the server and check whether some hypothetical event has occurred in a time period of interest. This process speeds up the actions as it can happen even in the absence of the vehicle and preserves the privacy of the car owner as no data, unrelated to an event, has to be decrypted as the relevance of some hidden data can be tested using SSE.

The first common characteristic feature of these use cases is that finding *all* the occurrences of a given keyword in the whole DB is not the goal. Particularly they highlight that finding some correspondence in a properly chosen part of the DB can also be meaningful. Secondly, new entries are frequently added to the DB and therefore rapid updates are

Manuscript received November 15, 2017; revised March 13, 2018. Date of publication March 15, 2018. Dr. sc. Toni Perković has been coordinating the review of this manuscript and approved it for publication.

Authors are with the Laboratory of Cryptography and Systems Security (CrySyS Lab), Department of Networked Systems and Services, Budapest University of Technology and Economics, Hungary (e-mails: mhorvath@crysys.hu, vajda@hit.bme.hu).

Digital Object Identifier (DOI): 10.24138/jcomss.v14i1.419

crucial, hitting the Achilles heel of the widespread inverted index approach.

A. Related Works

Several aspects of searchable encryption have been studied in the past years. We mention only a few of them and refer to recent surveys [4], [19] for an extensive summary on SSE. [7] captured first formally the intuitive goal of minimizing information leakage during keyword search (see section III-B). Schemes were put forward that allow not only single but also conjunctive/disjunctive keyword search [5]. Ranked keyword search over encrypted data was proposed by [20], [21]. Dynamic SSE schemes were designed to handle large DBs with possible updates (see details in subsection IV-D). Concurrently to our work [13] proposed a combination of the forward and inverted index approach. Keyword search in the public-key setting (PEKS) was introduced by [2] and later improved in several directions [4].

B. Our Contribution

In this revised and extended version of [10], we aim to build SSE that is optimized for the above scenarios and provably fulfils the strongest security requirements towards SSE schemes. While [10] presented a specific instance of such an SSE scheme (without a formal proof of security), we generalize the approach introduced in that work and propose a construction that is built from an arbitrary **IND-CPA** secure symmetric-key cipher and an **EU-CMA** secure MAC function working over asymmetric (Type-3) bilinear groups in which the Symmetric eXternal Diffie-Hellman (SXDH) assumption holds. Our modular approach allows for flexible choices of the underlying primitives helping the adoption of our scheme to concrete applications.

In order to address the above-described challenges, we return to the forward index method to design an SSE scheme that handles newly encrypted data without requiring any updates on the already stored DB. The additional benefit of this approach is that it allows periodical signing of the encrypted DB (that can be particularly useful in case of log files) ensuring that the DB was not modified, while the same is impossible in dynamically changing inverted indices. The core of our constriction is a keyword encryption and trapdoor generation method, both with randomized outputs that allows equality-test to determine whether a given trapdoor-ciphertext pair corresponds to the same keyword or not. These special keyword encryptions can then be attached to the ordinary encryption of a document (or file) and stored together on an honest but curious server. The security of our scheme against adaptive chosen-keyword attacks (**IND-CKA2**) is proven in the standard model.

The rest of the paper is organized as follows. In section II, we summarize the necessary background, while in section III the formal definition of the algorithms and security of SSE is given. Section IV is dedicated to the proposed scheme, its security analysis and comparisons with related concepts. Finally we conclude our findings in section V.

II. Preliminaries

First of all, we briefly introduce the underlying tools of our construction, namely bilinear maps and message authentication codes (MACs), and discuss our hardness assumption. However, we are going to make use of symmetric-key encryption (SE) in our construction, we omit further details as for our purposes the basic syntax suffices (we denote the encryption and decryption algorithms by **SE.Enc** and **SE.Dec** respectively and the used secret key by $\mathsf{sk}_{\mathrm{SE}}$). For more details on SE and the **IND-CPA** security notion, we refer to standard textbooks on cryptography.

A. Bilinear Pairings

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three multiplicative cyclic groups of prime order p. Let g_1 and g_2 be the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map (pairing), with the following properties:

1) Bilinearity: $\forall u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$
2) Non-degeneracy: $e(g_1, g_2) \neq 1$.

We say that $\mathcal{G} = \{p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e\}$ is a bilinear instance if all the group operations and the bilinear map e are efficiently computable. In this work, we apply the so-called "Type-3" pairings, meaning that $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable isomorphism exists between them. We note that based on both its efficiency and security, this pairing type is considered to be the ideal choice when instantiating a cryptosystem [6].

The security of our SSE scheme can be reduced to the hardness of the so-called Symmetric eXternal Diffie-Hellman (SXDH) problem that we define next. Informally speaking, given (g^a, g^b, g^c), where g is a generator of group $\mathbb{G}$ and $a, b, c \in \mathbb{Z}_p^*$, the Decisional Diffie-Hellman (DDH) problem is to decide whether $c = ab$ or not. The SXDH assumption states that no efficient algorithm can solve the DDH problem either in $\mathbb{G}_1$ or in $\mathbb{G}_2$ of a bilinear instance.

Assumption 1 (SXDH). *Let $g_i \in \mathbb{G}_i$ be a generator element of the group and $a_i, b_i \in \mathbb{Z}_p^*$ are uniformly random values for $i = 1, 2$. We say that the SXDH assumption holds in a bilinear instance $\mathcal{G}$ if given*

$$(g_i, g_i^{a_i}, g_i^{b_i}, g_i^{R_i}),$$

for $i = 1, 2$, no polynomial time algorithm can decide whether $R_i = a_i b_i$ or R_i is also a uniformly random value from $\mathbb{Z}_p^*$.

B. Message Authentication Codes

In our construction, we are going to make use of deterministic Message Authentication Codes (MAC) with a specific syntax to construct randomized MAC for our purposes.

Definition 1. *A deterministic (*randomized*) MAC consists of the following algorithms:*

MAC.KeyGen(λ)$\to$ $\mathsf{sk}_{\mathbf{MAC}}$ *This randomized algorithm generates secret key sk_{MAC} based on security parameter λ.*

MAC($\mathsf{sk}_{\mathbf{MAC}}, m$) $\to \tau$ *Using the secret key sk_{MAC}, this deterministic (*randomized*) algorithm produces a tag τ for the input message m.*

MAC.Verify$(\mathsf{sk}_{\mathbf{MAC}}, m, \tau) \rightarrow \{0,1\}$ *With the help of the secret key* sk_{MAC} *this deterministic algorithm checks whether* τ *was prepared using* m *or not.*

Note that in case of deterministic MACs, **MAC.Verify**$(\mathsf{sk}_{\text{MAC}}, m, \tau)$ simply computes **MAC**$(\mathsf{sk}_{\text{MAC}}, m) = \tau'$ and checks whether $\tau' = \tau$ or not.

In this work, we are interested in MACs in special form i.e., we assume that tag $\tau = \alpha^{F(\mathsf{sk}_{\text{MAC}}, m)}$, where α is a generator element of either group $\mathbb{G}_1$ or $\mathbb{G}_2$ of the pairing group $\mathcal{G}$ and F is some function of the secret key and the message to be authenticated. In the literature, several Pseudo-Random Functions (PRF) were described [16], [14], [3], [1] in the desired form under various hardness assumptions. However, for our purposes this stronger guarantee of pseudo-randomness is not required, any of these can serve as proper MAC functions satisfying the following requirement of existential unforgeability under chosen message attacks ($\mathbf{EU\text{-}CMA}$).

Definition 2 ($\mathbf{EU\text{-}CMA}$)**.** *We say that a MAC function (**MAC.KeyGen**, **MAC**, **MAC.Verify**) is secure if it is existentially unforgeable under chosen message attack (*$\mathbf{EU\text{-}CMA}$*) i.e., let* $\lambda \in \mathbb{N}$ *be a security parameter, and* $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_{q+1})$ *be an non-uniform adversary that has at most negligible advantage in the* $\mathbf{EU\text{-}CMA}_{MAC,\mathcal{A}}(\lambda)$ *experiment (depicted on Figure 1):*

$$\Pr(\mathbf{EU\text{-}CMA}_{MAC,\mathcal{A}}(\lambda) = 1) \leq \mathsf{negl}(\lambda).$$

```
$\mathbf{EU\text{-}CMA}_{MAC,\mathcal{A}}(\lambda)$ Security Game
$\mathsf{sk}_{\text{MAC}} \leftarrow\!\$\ \mathbf{MAC.KeyGen}(1^\lambda)$
$\{m_1, \mathsf{state}_{\mathcal{A}_1}\} \leftarrow \mathcal{A}_1(1^\lambda)$
$(\tau_1, m_1) \leftarrow \mathbf{MAC}(\mathsf{sk}_{\text{MAC}}, m_1)$
for $i = 2, \ldots, q$
    $\{m_i, \mathsf{state}_{\mathcal{A}_i}\} \leftarrow \mathcal{A}_i\left(1^\lambda, \mathsf{state}_{\mathcal{A}_{i-1}}, \{\tau_j, m_j\}_{j \in [i-1]}\right)$
    $(\tau_i, m_i) \leftarrow \mathbf{MAC}(\mathsf{sk}_{\text{MAC}}, m_i)$
$(\bar{\tau}, \bar{m}) \leftarrow\!\$\ \mathcal{A}_{q+1}\left(1^\lambda, \mathsf{state}_{\mathcal{A}_q}, \{(\tau_i, m_i)\}_{i \in [q]}\right)$
if $\mathbf{MAC.Verify}(\mathsf{sk}_{\text{MAC}}, \bar{m}, \bar{\tau}) = 1$ and $\bar{m} \notin \{m_1, \ldots, m_q\}$
$\mathbf{return}\ 1$
```

Figure 1. $\mathbf{EU\text{-}CMA}$ security game.

For instance, the construction of [16] can be used assuming the hardness of the DDH problem that is implied by the SXDH assumption used in this work.

III. Searchable Symmetric Encryption

The focus of this work is SSE, more precisely using the terminology of [4] we are interested in the single writer/single reader setting. In an abstract version of the scenarios, described in the Introduction, the data owner generates the system parameters, secret keys, encrypts data and prepares the trapdoors using the secret keys. Besides storing the ciphertexts, the server is able to search over encrypted data using trapdoors for specific keywords, issued by the data owner. The server is assumed to be semi-trusted (honest-but-curious) and the communication channel between the user and the server supposed to be authenticated. Next, we define the algorithms and the security of an SSE scheme that fits into this abstract scenario.

A. Definition of SSE

An SSE scheme consists of the following algorithms.

SSE.Setup$(\lambda) \rightarrow (\mathsf{P}, \mathsf{sk})$ Upon receiving a security parameter λ it outputs the system parameters P and a secret key sk.

SSE.Enc$(\mathsf{P}, \mathsf{sk}, m, w) \rightarrow (C)$ Using the secret key sk it computes ciphertext C that encrypts m under keyword w.

SSE.TrpdGen$(\mathsf{P}, \mathsf{sk}, \hat{w}) \rightarrow (T)$ Using the secret key sk it computes a trapdoor T that can be used to test whether some ciphertext C was encrypted under keyword $\hat{w}$ or not.

SSE.Dec$(\mathsf{P}, \mathsf{sk}, C) \rightarrow (m)$ It decrypts ciphertext C with secret-key sk and outputs the resulting plaintext m.

SSE.Test$(\mathsf{P}, T, C) \rightarrow \{0,1\}$ The equality testing algorithm outputs 1 if T and C encodes the same keyword i.e., $w = \hat{w}$ and 0 otherwise.

B. Security Model for SSE

The commonly used security model for SSE was defined by [7] to capture the intuition that, in the course of using the scheme, the remotely stored files and search queries together do not leak more information about the underlying data than the search pattern and the search outcome. In our security definition, we follow [7] but we formulate it – to the best of our knowledge for the first time – in the context of a forward index.

While in the inverted index-based approach the index and the ciphertexts are handled separately, in our case of a forward index, it is natural to view the "index" as part of the ciphertext. We formulate the model in this way, defining indistinguishability under adaptive chosen keyword attack ($\mathbf{IND\text{-}CKA2}$) through a game between a challenger and an adversary. In the game, the adversary has to recognize which one of two challenge datasets (consisting of messages and their keywords chosen by herself) was encrypted by the challenger. While we are interested in a setting where the database is not static but can be expanded dynamically, in the security model we still restrict the adversary to query encryptions once and query only trapdoors adaptively. The reason behind this is that (in both of our motivating scenarios) searchability in a subset of the DB can be enforced by choosing time-dependent keywords (i.e., "error-01-01-2018" instead of "error"). With the natural assumption that searches only apply to time periods that are already over (and thus updates does not affect them) without loss of generality, we can restrict our attention to one specific – now static – subset of the DB (i.e., data from a given time frame). Note that in a forward index even the knowledge of the order of ciphertexts can help the attacker, that is why our challenger provides her with a random permutation of ciphertexts prepared from the randomly chosen challenge message set. The adversary has access not only to the encryptions themselves but also to a trapdoor generation oracle that can be queried adaptively with pairs of keywords corresponding to the two challenge sets. The oracle answers consistently with a trapdoor for that keyword which belongs to the encrypted challenge data set. The only restriction is that the queried keywords cannot separate the two challenge sets, as we are interested in information leakage beyond the search result.

For the ease of exposition, we assume that there is a single keyword for each message, but this can be easily generalized. More formally we use the subsequent definition of security following [7, §4.2.2].

Definition 3 (Adaptive indistinguishably)**.** *Let SSE = (**Setup**, **Enc**, **TrpdGen**, **Dec**, **Test**) be a secret-key searchable encryption scheme, $\lambda \in \mathbb{N}$ a security parameter, and $\mathcal{A} = (\mathcal{A}_0, \ldots, \mathcal{A}_{q+1})$ a non-uniform adversary. Consider the probabilistic experiment $\mathbf{Ind\text{-}CKA2}_{\mathbf{SSE},\mathcal{A}}(\lambda)$ depicted on Figure 2 with the restriction that the number of keyword matches between the challenge message sets and the corresponding test-key queries are equal i.e.,*

$$\#\{i|\hat{w}_j^0 = w_i^0 \textit{ for } j \in [k]\} = \#\{i|\hat{w}_j^1 = w_i^1 \textit{ for } j \in [k]\}$$

for all $k = 1, \ldots, q$, where q is some polynomial of the security parameter λ. We say that an SSE scheme is secure in the sense of adaptive indistinguishability if for all polynomial-time adversaries $\mathcal{A} = (\mathcal{A}_0, \ldots, \mathcal{A}_{q+1})$,

$$\Pr(\mathbf{Ind\text{-}CKA2}_{\mathbf{SSE},\mathcal{A}}(\lambda) = 1) \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

```
$\mathbf{Ind\text{-}CKA2}_{\mathbf{SSE},\mathcal{A}}(\lambda)$ Security Game
$\mathsf{sk} \leftarrow_\$ \mathbf{SSE.Setup}(1^\lambda)$
$b \leftarrow_\$ \{0,1\}$
$(\mathsf{state}_{\mathcal{A}_0}, D^0, D^1) \leftarrow \mathcal{A}_0(1^\lambda)$
parse $D^b$ as $\{(m_1^b, w_1^b), \ldots, (m_n^b, w_n^b)\}$
for $1 \leq j \leq n$,
    $C_i^b \leftarrow_\$ \mathbf{SSE.Enc}(\mathsf{sk}, m_i^b, w_i^b)$,
$C^b := (C_{\pi_1}^b, \ldots, C_{\pi_n}^b)$ for a random permutation $\pi$,
for $1 \leq j \leq q$,
    $(\mathsf{state}_{\mathcal{A}_j}, \hat{w}_j^0, \hat{w}_j^1) \leftarrow \mathcal{A}_j(\mathsf{state}_{\mathcal{A}_{j-1}}, C^b, \{T_i^b\}_{i\in[j]})$
    $T_j^b \leftarrow_\$ \mathbf{SSE.TrpdGen}(\mathsf{sk}, \hat{w}_j^b)$
$b' \leftarrow \mathcal{A}_{q+1}(\mathsf{state}_{\mathcal{A}_q}, C^b, \{T_j\}_{i=1,\ldots,q})$
**return** $b = b'$
```

Figure 2. **IND-CKA2** security game for forward index SSE schemes.

IV. Proposed Scheme

In this part, we first describe a randomized MAC function based on any deterministic one in the form described in subsection II-B and then utilize the resulting MAC to build our new SSE algorithms. We analyse both the security and the performance of the proposed scheme in subsection IV-C and IV-D respectively.

A. Randomized MAC for SSE

In our SSE construction, we are going to make crucial use of two properties of the used MAC function. First, it needs to work over a bilinear pairing group i.e., depending on its input, it should prepare the output tag in one of the groups $\mathbb{G}_1$, $\mathbb{G}_2$. It also needs to be randomized meaning that a verification algorithm needs to be explicitly defined, as preparing the MAC of the same message two times results in different tags. We propose a simple solution, denoted by MAC', fulfilling these requirements, based on any deterministic MAC described in subsection II-B.

MAC.KeyGen'$(\lambda, \mathcal{G})$→ $\mathsf{sk}_{\mathbf{MAC}}$ It is identical to the original **MAC.KeyGen**(λ).

MAC'$(g_i, \mathsf{sk}_{\mathbf{MAC}}, m) \rightarrow \tau$ On input g_i, which is a generator element of group $\mathbb{G}_i$, the output is prepared in this group. In order to do so, the algorithm first samples a random $r \in \mathbb{Z}_p^*$ and sets $\alpha = g_i^r$. Then using α, it computes $\mathbf{MAC}(\mathsf{sk}_{\mathrm{MAC}}, m) = \alpha^{F(\mathsf{sk}_{\mathrm{MAC}}, m)} := \tau'$. The output is the two-tuple $\tau = (\alpha, \tau')$.

MAC.Verify'$(\mathsf{sk}_{\mathbf{MAC}}, m, \tau) \rightarrow \{0, 1\}$ It takes α from τ and checks if $\alpha^{F(\mathsf{sk}_{\mathrm{MAC}}, m)} = \tau'$. If so it outputs 1, otherwise 0.

The **EU-CMA** security of the this randomized MAC follows from the security of the underlying deterministic MAC.

B. SSE Construction

The intuition behind the construction of our **Test** algorithm is fairly simple. We build the trapdoors for a specific keyword and the keyword related ciphertext components in a symmetric manner: both are randomised MACs of the underlying keyword, however, represented in distinct groups $\mathbb{G}_1$ or $\mathbb{G}_2$. This enables us to test equality by "mixing" the ciphertext and the trapdoor in two different ways (using the pairing operation) that are equal only if the underlying keywords are the same. Using distinct groups prevents the testability both among ciphertexts and among trapdoors. In more detail, the algorithms are the following.

SSE.Setup$(\lambda) \rightarrow (\mathsf{P}, \mathsf{sk})$ It proceeds with the following steps:

- samples an instance of Type-3, asymmetric bilinear pairing groups, $\mathcal{G} = \{p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e\}$,
- chooses an instance of the above defined MAC' that implicitly defines function $F : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$,
- runs the **MAC.KeyGen**' algorithm of the chosen MAC function to generate $\mathsf{sk}_{\mathrm{MAC}}$,
- samples secret keys $\mathsf{sk}_{\mathrm{SE}}$ for the used SE scheme,
- samples secret key $\mathsf{sk}_{\mathrm{SSE}} \leftarrow_\$ \mathbb{Z}_p^*$,
- outputs the public parameters $\mathsf{P} = (\mathcal{G}, F, g^{\mathsf{sk}_{\mathrm{SSE}}})$ and the secret key $\mathsf{sk} = \{\mathsf{sk}_{\mathrm{MAC}}, \mathsf{sk}_{\mathrm{SE}}, \mathsf{sk}_{\mathrm{SSE}}\}$.

SSE.Enc$(\mathsf{P}, \mathsf{sk}, m, w) \rightarrow (C)$ The encryption of data m is executed using the encryption algorithm of a symmetric-key cipher while for the encryption of the keyword we utilize first MAC':

- it runs **MAC**'$(g_1, \mathsf{sk}_{\mathrm{MAC}}, w) = (\alpha = g_1^r, \tau' = g_1^{rF(\mathsf{sk}_{\mathrm{MAC}}, w)})$ where $r \in \mathbb{Z}_p^*$ is sampled randomly,
- finally it computes the following ciphertext:

$$C = \left(c_1 = \alpha, c_2 = \tau'^{\mathsf{sk}_{\mathrm{SSE}}}, c_3 = \mathbf{SE.Enc}(\mathsf{sk}_{\mathrm{SE}}, m)\right).$$

SSE.TrpdGen$(\mathsf{P}, \mathsf{sk}, \hat{w}) \rightarrow (T)$ Upon receiving a keyword $\hat{w}$

- it runs **MAC**'$(g_2, \mathsf{sk}_{\text{MAC}}, \hat{w}) = (\alpha = g_2^{r'}, \tau' = g_2^{r' F(\mathsf{sk}_{\text{MAC}}, \hat{w})})$ where $r' \in \mathbb{Z}_p^*$ is sampled randomly,
- and computes the SSE trapdoor in the following form:

$$T = \left(t_1 = \alpha, t_2 = \tau'^{\mathsf{sk}_{\text{SSE}}}\right).$$

SSE.Dec$(\mathsf{sk}, C) \rightarrow (m)$ After parsing C as (c_1, c_2, c_3), in order to recover the encrypted data the decryption algorithm of the symmetric-key scheme is used and **SE.Dec**$(\mathsf{sk}_{\text{SE}}, c_3) = m$ is returned, while the rest of the ciphertext is not affected.

SSE.Test$(\mathsf{P}, T, C) \rightarrow \{0, 1\}$ To test whether a ciphertext C (parsed as (c_1, c_2, c_3)) was encoded using the same keyword that is hidden in trapdoor T (parsed as (t_1, t_2)) the following equality is checked:

$$e(c_2, t_1) = e(c_1, t_2).$$

If the equality holds the output is 1, otherwise 0.

The correctness of the **SSE.Dec** and **SSE.Test** algorithms follows after substitution of the proper values into the formulas i.e., in case of the latter one $e(c_2, t_1) = e(g_1, g_2)^{rr'\mathsf{sk}_{\text{SSE}} F(\mathsf{sk}_{\text{MAC}}, w)} = e(g_1, g_2)^{rr'\mathsf{sk}_{\text{SSE}} F(\mathsf{sk}_{\text{MAC}}, \hat{w})} = e(c_1, t_2)$ iff $w = \hat{w}$.

C. Security Analysis

In this part, we formulate our main theorem and its proof.

Theorem IV.1. *If the used symmetric-key encryption scheme is* $\mathbf{IND\text{-}CPA}$ *secure, the underlying MAC function is* $\mathbf{EU\text{-}CMA}$ *secure and the SXDH assumption holds in the pairing group* $\mathcal{G}$*, then the proposed SSE scheme is* $\mathbf{IND\text{-}CKA2}$ *secure according to Definition 3.*

Before proving the theorem, we prove three lemmas that are going to be used in the proof of the theorem.

Lemma IV.2. *The SSE ciphertext* $C = (c_1, c_2, c_3)$ *alone (when no trapdoor is issued) is semantically (*$\mathbf{IND\text{-}CPA}$*) secure, if the underlying SE scheme is* $\mathbf{IND\text{-}CPA}$ *secure and the SXDH assumption holds.*

Proof sketch. By our assumption, the ciphertext of the symmetric-key cipher is semantically secure and thus in order to prove the lemma we have to show that the other two components c_1, c_2 of the SSE ciphertext is also indistinguishable from truly random values (when trapdoors do not help to distinguish them). Note that the structure of these values is reminiscent of ElGamal ciphertexts. The difference is that the "message" $F(\mathsf{sk}_{\text{MAC}}, w)$ is not multiplied by the randomizing factor $g_i^{r\mathsf{sk}_{\text{SSE}}}$ but exponentiated to it, however, the same reduction of the DDH assumption in $\mathbb{G}_1$ to the security of the scheme works here just like in case of the ElGamal cryptosystem. In the reduction the simulator answers for queries for keywords w_j (for some j, smaller than the allowed number queries) with $c_1 = g_1^{r_j}, c_2 = g_1^{r_j b_1 F(\mathsf{sk}_{\text{MAC}}, w_j)}$, while in the challenge phase sends $c_1 = g_1^{a_1}, c_2 = g_1^{R \cdot F(\mathsf{sk}_{\text{MAC}}, w_b)}$ for randomly chosen $b \in \{0, 1\}$. Finally the output of the simulator is 1 if the guess b' of the adversary is $b = b'$ and 0 otherwise. □

Lemma IV.3. *As long as the SXDH assumption holds and testing equality with ciphertexts does not help distinguishing the SSE trapdoor* $T = (t_1, t_2)$ *from a truly random tuple,* T *hides the underlying keyword with* $\mathbf{IND\text{-}CPA}$ *security.*

Proof sketch. Note that, just like c_1, c_2 from C, the structure of T also resembles the ElGamal ciphertext and thus the proof of the lemma again follows the blueprint of reducing the security of the ElGamal cryptosystem to the DDH assumption (i.e., in our case, to the SXDH assumption in group $\mathbb{G}_2$). □

Lemma IV.4. *The SSE trapdoor* $T = (t_1, t_2)$ *is existentially-unforgeable (*$\mathbf{EU\text{-}CMA}$ *secure).*

Proof. We are going to show that if there exists an efficient algorithm $\mathcal{A}$ that can forge trapdoor $T = (t_1, t_2)$, then there also exist $\mathcal{B}$ that using $\mathcal{A}$ can forge the underlying MAC' function, contradicting with its $\mathbf{EU\text{-}CMA}$ security.

According to Definition 2, $\mathcal{B}$ has access to a CMA oracle which replies with $\tau = (\alpha, \tau')$ for a message request w. This allows $\mathcal{B}$ to answer the trapdoor requests of $\mathcal{A}$ for any keyword w with a valid trapdoor $T = (t_1 = \alpha = g_2^{r'}, t_2 = \tau'^{\mathsf{sk}_{\text{SSE}}} = g_2^{r'\mathsf{sk}_{\text{SSE}} F(\mathsf{sk}_{\text{MAC}}, w)})$, after generating an SSE secret key $\mathsf{sk}_{\text{SSE}} \in \mathbb{Z}_p^*$. When $\mathcal{A}$ outputs the forged trapdoor $\bar{T} = \left(\bar{t_1} = g_2^{\bar{r}}, \bar{t_2} = g_2^{\bar{r}\mathsf{sk}_{\text{SSE}} F(\mathsf{sk}_{\text{MAC}}, \bar{w})}\right)$ for keyword $\bar{w}$, that was not requested previously, $\mathcal{B}$ can compute $\bar{\tau} = (\bar{\alpha} = t_1, \bar{\tau'} = \bar{t_2})^{1/\mathsf{sk}_{\text{SSE}}}$ and output the pair $(\bar{\tau'}, \bar{w})$ as the forged message-tag pair, breaking the security of the underlying MAC'. □

Now we are ready to prove our main theorem.

Proof of Theorem IV.1. We are going to define several hybrid games where in the last hybrid the attacker receives encryptions of random values under random keywords instead of the challenge message and trapdoors for random keywords instead of the queried ones, thus she cannot have a non-negligible advantage in that game. In order to prove the theorem, we systematically show that the subsequent hybrids are indistinguishable for the adversary and thus her advantage in the original game is essentially negligible as well.

For the ease of exposition we introduce some notations. Let $\vec{W} = (W_1, \ldots, W_m)$ be the vector of all queried keyword values either for a trapdoor or as part of the encryption query (thus $m \leq n + q$). We denote the set of message (keyword) query indices that match with W_j by $I_{W_j} = \{i | w_i^b = W_j\}$ (and by $\hat{I}_{W_j} = \{i | \hat{w}_i^b = W_j\}$ respectively). Let R be an initially empty list of two-tuples (W_i, R_i) in which the challenger can store random R_i values for each W_i. Finally we denote the components of C_i^b by $c_{1,i}, c_{2,i}, c_{3,i}$ and the components of T_i^b by $t_{1,i}, t_{2,i}$. We define the following hybrid games.

Hyb_0: it corresponds to the original game.

Hyb_1: the same as Hyb_0 but in C^b, the third ciphertext components $c_{3,i}$ are substituted with random values of the ciphertext space for all $i \in [n]$.

$\mathsf{Hyb}_{2,1}$: the same as Hyb_1, with the following exception. A random $R_1 \leftarrow\!\!\$\, \mathbb{Z}_p^*$ is sampled and (W_1, R_1) is appended to R. For each $i \in I_{W_1}$ and the challenger answers with C_i^b in which the second parameter is $c_{2,i} = g_1^{r_i R_1}$ instead of $c_{2,i} = g_1^{r_i \mathsf{sk}_{\mathrm{SSE}} F(\mathsf{sk}_{\mathrm{MAC}}, w_i^b)}$. Similarly for each $i \in \hat{I}_{W_1}$ the challenger answers to trapdoor queries for $\hat{w}_i^b$ with T_i^b in which $t_{2,i} = g_2^{r'_i R_1}$ instead of $t_{2,i} = g_2^{r'_i \mathsf{sk}_{\mathrm{SSE}} F(\mathsf{sk}_{\mathrm{MAC}}, \hat{w}_i^b)}$.

$\vdots$

$\mathsf{Hyb}_{2,m}$**:** the same as $\mathsf{Hyb}_{2,m-1}$, with the following exception. A random $R_m \leftarrow\!\!\$\, \mathbb{Z}_p^*$ is sampled and (W_m, R_m) is appended to R. For each $i \in I_{W_m}$ the challenger answers with C_i^b in which the second parameter is $c_{2,i} = g_1^{r_i R_m}$ instead of $c_{2,i} = g_1^{r_i \mathsf{sk}_{\mathrm{SSE}} F(\mathsf{sk}_{\mathrm{MAC}}, w_i^b)}$. Similarly for each $i \in \hat{I}_{W_m}$ the challenger answers to trapdoor queries for $\hat{w}_i^b$ with T_i^b in which $t_{2,i} = g_2^{r'_i R_m}$ instead of $t_{2,i} = g_2^{r'_i \mathsf{sk}_{\mathrm{SSE}} F(\mathsf{sk}_{\mathrm{MAC}}, \hat{w}_m^b)}$.

Now we are going to show that the subsequent hybrids are indistinguishable for the attacker.

Claim IV.5. *The computational indistinguishability of* Hyb_0 *and* Hyb_1 *follows from Lemmas IV.2 and IV.3.*

Lemma IV.2 states that $c_{3,i}$ is semantically secure or in other words that the adversary can have at most negligible advantage in distinguishing it from a random value. At the same time, keywords are possibly not independent of the data and thus the other ciphertext components $c_{1,i}, c_{2,i}$ can help $\mathcal{A}$to recognize the shift between the hybrids. However, even if $w_i = f(m_i)$ for some function f, the testing algorithm cannot help $\mathcal{A}$ because the results of it are identical in the two hybrids. The only remaining possibility is that $c_{1,i}, c_{2,i}$ leaks information about w_i (and thus indirectly about m_i) but this would contradict with Lemma IV.3.

Consequently, $\mathcal{A}$ can have at most $2n + q$ times negligible advantage in distinguishing Hyb_0 and Hyb_1, that is still negligible.

Claim IV.6. Hyb_1 *and* $\mathsf{Hyb}_{2,1}$ *are computationally indistinguishable because of Lemmas IV.2, IV.3 and IV.4.*

First of all, we notice that the **SSE.Test** algorithm does not help the adversary in distinguishing these hybrids as it clearly has the same outputs in both cases. Also note that semantic security of the ciphertexts and the trapdoors (guaranteed by Lemmas IV.2, IV.3) are necessary but not sufficient to imply indistinguishability. The reason is that in Hyb_1, $\mathcal{A}$ might be able to use the trapdoor queries to gather valid keyword-trapdoor pairs adaptively in order to generate a valid trapdoor T_{q+1} for keyword $\hat{w}_{q+1}$ that was not queried in the game, thus it might not fulfil the constraints of the game[1]. However, according to Lemma IV.4, the trapdoors are **EU-CMA** secure meaning that $\mathcal{A}$ can have only negligible advantage even in Hyb_1 to generate a non-queried keyword-trapdoor pair just like in $\mathsf{Hyb}_{2,1}$ in which the trapdoor is independent of the actual keyword. This shows that the attacker can have at most $(|I_{W_1} + \hat{I}_{W_1}| + 1)$ times negligible advantage in distinguishing Hyb_1 and $\mathsf{Hyb}_{2,1}$.

Note that the indistinguishability of the remaining hybrids follows from an analogous argument (as we repeat the same steps in those cases) showing that the following claim holds as well.

Claim IV.7. *The computational indistinguishability of* $\mathsf{Hyb}_{2,i-1}$ *and* $\mathsf{Hyb}_{2,i}$ *for any* $i \in [2, m]$ *follows from Lemmas IV.2, IV.3 and IV.4.*

By showing the indistinguishability of the hybrids we have proven the theorem. □

Forward privacy guarantees that trapdoors can only be used to test keywords of documents which were already part of the DB at the time of issuing the trapdoor. We remark that by default, our scheme is not forward secure (trapdoors can be stored by the server and can be used to test future ciphertexts) but the already mentioned usage of time-dependent keywords can remedy this deficiency, thus achieving forward (and also backward) privacy between the time periods. Note that the same solution of time-specific keywords would result in an infinitely growing inverted index.

D. Evaluation and Comparisons

We compare our results with dynamic SSE schemes which are the most suitable in the literature for the use cases that we considered in this work. Table I shows a comparison using the following notations: n denotes the number of documents (data entries), w_D is the number of keywords per a specific document, W is the total number of distinct keywords in the DB, n_w is the number of documents matching the searched keyword w, a is the total number of additions to the DB and d is the total number of deletions, b is the bit length of encrypted documents. * indicates that update requires some rounds of interaction between the server and the client and ** denotes amortized complexity.

As predicted in the introduction, Table I confirms that our search strategy of sequential scan is not competitive, unless only a small portion of the DB (e.g., at most $(\log W)/w_D$ ciphertexts) is enough to scan, that is realistic in the investigated scenarios. The most important benefits of our scheme include resistance against adaptive chosen-keyword attacks in the standard model and non-interactive update of ciphertexts with low complexity, depending only on the number of keywords. Moreover, our ciphertexts (including the index) and trapdoors are very short, consisting of $w_D + 2$ and 2 group elements respectively.

Let us emphasize that updating the DB with a new record is straightforward in our approach. The client encrypts the data together with the keywords and the server only has to store the received ciphertexts contrary to other solutions where the server has to "find the place" of the new entry in the index. This latter process also harms the privacy of updates in most cases by leaking information about the added keywords (e.g., all documents with common keywords can be identified). In our case, only the number of added keywords is leaked, however, in the targeted applications it is plausible to assume that the number of keywords is not varying among the different "documents".

[1] We note that in case of Claim IV.5 the same was not a problem, as the extra knowledge of another valid message-ciphertext pair would not help the attacker in the game.

Table I
COMPARISON OF OUR RESULTS AND DYNAMIC SSE SCHEMES. (FOR NOTATIONS AND EVALUATION SEE SUBSECTION IV-D.)

Scheme	Model	Security	Fw/Bw Privacy	Update Complexity	Update Privacy	Search Complexity
[17]	Standard	**IND-CPA**	× / ×	$O(b)$	×	$O(n \cdot b)$
[15]	Standard	**IND-CKA2**	× / ×	$O(w_D)^*$	×	$O(\log W)$
[12]	ROM	**CKA2**	× / ×	$O(w_D)$	×	$O(n_w)$
[11]	ROM	**CKA2**	× / ×	$O(\log n)^*$	✓	$O(n_w \log n)$
[5]	ROM	**CKA2**	× / ×	$O(w_D + W \log n)$	×	$O(n_w + a + d)$
[18]	ROM	**CKA2**	✓ / ×	$O(w_D \log(nW))^*$	✓	$O(n_w + d)$
[9]	ROM	**CKA2**	× / ×	$O(nw_D/D)^{**}$	×	$O(nw_D/D)^{**}$
[22]	ROM	**CKA2**	✓ / ✓	$O(W)$	✓	$O(W)$
[8]	Standard	**IND-CKA2**	× / ×	$O(w_D \cdot W)$	×	$O(\log n)$
[13]	ROM	**CKA2**	✓ / ×	$O(w_D)^*$	✓	$O(n_w)^*$
Our scheme	Standard	**IND-CKA2**	✓ / ✓	$O(w_D)$	✓	$O(n \cdot w_D)$

V. CONCLUSION

In this work, we revisited the role of sequential scan in searchable encryption and showed a construction that outperforms the existing solutions in real-life scenarios when not the entire database is scanned. Our scheme was proven **IND-CKA2** secure in the standard model assuming the widely used SXDH holds. To the best of our knowledge, this is the first scheme with a forward index that is proven secure in this strong model. Our modular design allows flexible choices of the underlying primitives in a future implementation.

ACKNOWLEDGMENT

The authors would like to thank the valuable remarks of Levente Buttyán. The research presented in this paper was supported by the National Research, Development and Innovation Office – NKFIH, under grant contract no. 116675 (K).

REFERENCES

[1] Abdalla, M., Benhamouda, F., and Passelègue, A. An algebraic framework for pseudorandom functions and applications to related-key security. In *CRYPTO 2015, Proceedings, Part I*, pages 388–409.

[2] Boneh, D., Crescenzo, G. D., Ostrovsky, R., and Persiano, G. Public key encryption with keyword search. In *EUROCRYPT 2004, Proceedings*, pages 506–522.

[3] Boneh, D., Montgomery, H. W., and Raghunathan, A. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In *Proceedings of the 17th ACM CCS 2010*, pages 131–140.

[4] Bösch, C., Hartel, P. H., Jonker, W., and Peter, A. A survey of provably secure searchable encryption. *ACM Comput. Surv. 2014*, 47(2):18:1–18:51.

[5] Cash, D., Jaeger, J., Jarecki, S., Jutla, C. S., Krawczyk, H., Rosu, M., and Steiner, M. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014*.

[6] Chatterjee, S. and Menezes, A. On cryptographic protocols employing asymmetric pairings – the role of Ψ revisited. *Discrete Applied Mathematics 2011*, 159(13):1311–1322.

[7] Curtmola, R., Garay, J. A., Kamara, S., and Ostrovsky, R. Searchable symmetric encryption: improved definitions and efficient constructions. In Juels, A., Wright, R. N., and di Vimercati, S. D. C., editors, *Proceedings of the 13th ACM CCS, 2006*, pages 79–88. ACM.

[8] Gajek, S. Dynamic symmetric searchable encryption from constrained functional encryption. In *Topics in Cryptology – CT-RSA 2016*,p. 75–89.

[9] Hahn, F. and Kerschbaum, F. Searchable encryption with secure and efficient updates. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2014*, pages 310–320.

[10] Horváth, M. and Vajda, I. Searchable Symmetric Encryption: Sequential Scan Can Be Practical. *The 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2017).*

[11] Kamara, S. and Papamanthou, C. Parallel and dynamic searchable symmetric encryption. In *Financial Cryptography and Data Security – FC 2013, Revised Selected Papers*, pages 258–274.

[12] Kamara, S., Papamanthou, C., and Roeder, T. Dynamic searchable symmetric encryption. In *the ACM Conference on Computer and Communications Security, CCS'12*, pages 965–976.

[13] Kim, K. S., Kim, M., Lee, D., Park, J. H. and Kim, W. Forward Secure Dynamic Searchable Symmetric Encryption with Efficient Updates In *Proceedings of the ACM CCS 2017*, pages 1449–1463.

[14] Lewko, A. B. and Waters, B. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In *Proceedings of the ACM CCS 2009*, pages 112–120.

[15] van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P. H., and Jonker, W. Computationally efficient searchable symmetric encryption. In *Secure Data Management, 7th VLDB Workshop, SDM 2010, Proceedings*, pages 87–100.

[16] Naor, M. and Reingold, O. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97*, pages 458–467.

[17] Song, D. X., Wagner, D., and Perrig, A. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy, 2000*, pages 44–55.

[18] Stefanov, E., Papamanthou, C., and Shi, E. Practical dynamic searchable encryption with small leakage. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014.*

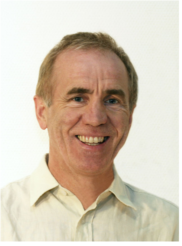
[19] Poh, G. S., Chin, J., Yau, W., Choo, K. R. and Mohamad, M. S. Searchable Symmetric Encryption: Designs and Challenges. ACM Comput. Surv. 50, 3, Article 40 (May 2017), 37 pages.

[20] Xia, Z., Wang, X., Sun, X., and Wang, Q. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Trans. Parallel Distrib. Syst. 2016*, 27(2):340–352.

[21] Yang, Y., Li, H., Liu, W., Yao, H., and Wen, M. Secure dynamic searchable symmetric encryption with constant document update cost. In *IEEE GLOBECOM 2014*, pages 775–780.

[22] Yavuz, A. A. and Guajardo, J. Dynamic searchable symmetric encryption with minimal leakage and efficient updates on commodity hardware. In *Selected Areas in Cryptography – SAC 2015, Revised Selected Papers*, pages 241–259.

Máté Horváth obtained his MSc diploma in computer science in the Security and Privacy program of EIT Digital at the University of Trento (Italy) and Eötvös Loránd University (Hungary). His bachelor degree is in mathematics from the Technical University of Budapest where he is pursuing a PhD currently. He has been doing research in the CrySyS Lab under the guidance of prof. Levente Buttyán since 2014.



István Vajda graduated from the Telecommunication Department at the Technical University of Budapest. He received the PhD and DSc degrees in 1985 and 1997, respectively. Since 1998, he has been a Professor at the Department of Informatics. During 1990's his research interest was in algebraic code designs for secure multiple access channels. Recently, his research interests are in design and analysis of secure systems, with a special emphasis on provably secure cryptographic primitives and protocols. His application expertise covers secure wireless communication, secure routing and sensor networks.