Verifiable Symmetric Searchable Encryption for FOIA compliance

ECE 407: Cryptography, UIUC

Advisor: Prof. Andrew Miller
Authors: Monica Pardeshi, Shengyu Ge, Milind Kumar V

Contents

List of Figures

Abstract

The Freedom of Information Act [3] (FOIA) allows citizens of a democracy to request access to documents- emails, memos, voice messages and so forth- concerning the functioning of the government. In this work, we address the problem of retrieving all the emails of a user- presumably an employee at a public institution obligated to respond to FOIA requests- containing a specific keyword while preserving the privacy of their emails. This is done using Verifiable Searchable Symmetric Encryption (VSSE). SSE schemes ensure that a curious Cloud Service Provider (CSP) learns nothing about the stored emails by suitably encrypting their plaintexts while maintaining the capability to search the stored data for specific tokens. The verifiability ensures that a CSP does not suppress a part of query response which is crucial to the FOIA setting. The result of our work is an open-source Python application that implements the desired VSSE scheme.

1 Introduction

Under the Freedom of Information Act (FOIA) in the United States of America [4], [3] the Government is required to disclose, partially or completely, documents previously controlled by it upon request. This process is crucial to the democratic functioning of the state and applies to most files, documentation and correspondence - emails, voice messages, etc. generated during its operation. In this work we will look at the specific task of retrieving all emails that contain a given word, called keyword henceforth, presumably originating from a FOIA query, under the assumption that all communication is end to end encrypted. For example, this could involve returning all the emails containing the word "contract" in them upon request.

Without the assumption of end to end encryption, a naive method to retrieve emails containing a queried word would be to have the email server search through the plaintexts of the emails sent by a user and return the relevant ones. However, this requires that the server have access to the plaintexts of all the emails sent by the user. This is not desirable. When communications are encrypted, the server only sees the ciphertexts of the messages generated by the user but is unable to retrieve the messages (rather, their ciphertexts) containing the queried word. A final method to return all such emails is to have all employees manually search through their emails for the specified word but this is extremely inefficient, prone to errors and does not preclude the possibility of emails being hidden with malicious intent. This problem can be solved using Searchable Symmetric Encryption (SSE) which devises schemes to allow for storage of data on cloud servers while retaining the ability to return the messages which contain a specific keyword. In particular, the SSE scheme used will need to be verifiable which in our application setting translates to the CSP having to return exactly those results which correspond to the query made. This prevents a CSP from hiding some results or even from claiming an unrelated result as corresponding to the query made.

This document is organized as follows. Section 2 discusses relevant work. Section 4 presents a discussion of the system model, the relevance of VSSE and the scope of this work. Section 5 discusses the implementation details of the algorithm and Section 6 introduces the relevant security notions and also provides a brief security analysis. Finally, Section 7 describes the possible extensions.

2 Related work

[2] introduces verifiable and dynamic SSE schemes which are central to this work. The dynamic scheme allows for the addition and deletion of documents while still supporting search outcome verifiability. [5] introduces the notion of a Semi-Honest But Curious CSP which is the threat model we use in this work. Further, the IND-CKA2 security notion and its modifications are discussed in [6].

3 Our contributions

Several implementations of SSE exist, for example [7], [8], [1]. However, most of these are not verifiable, including [7], [8]. In addition, our implementation will be secure for the SHBC-CSP model, a security notion that was introduced in [5]. Our reference, [2], was also one of the first to use bitmap based indexes for more space efficient encryption. The first was [9], but this scheme was also not verifiable. Our implementation will combine verifiability and space efficiency. [1] is indeed verifiable, but as Figure 1 (from Zhang's repo) shows, the way it verifies the result is simply using the search time as a proof. Unike [1], our reference [2] uses the MAC that generates TAG and checks TAG. Therefore, this implementation would be more complex and secure.

4 Setting

We consider a simplified version of the aforementioned problem and discuss a solution based on [2]. This work focuses on the static case- where the number of emails and query words is fixed and implement the verifiable SSE scheme from [2]. This assumes that all communication that can happen has already happened and the only goal is to provide privacy preserving searchable encryption. Considering a dynamic setting is beyond the scope of this work and can be done using the dynamic VSSE scheme discussed in [2].

This setting is described in greater detail below

core code

```
gen_update_token(){
    sc = get_search_time(w);
    tw = gen_enc_token(w);
    uc = get_update_time(w);
    l = Util::H1(tw + std::to_string(uc + 1));
    e = Util::Xor(op + ind, Util::H2(tw + std::to_string(uc + 1)));
    set_update_time(w, uc + 1);
    // search_time is used to store the proof
    set_search_time(w, Xor(sc, ind));
}

gen_search_token(){
    tw = gen_enc_token(w);
    uc = get_update_time(w);
}

search() {
    for(i=uc to 1){
        u = H1(tw + uc);
        e = get(u);
        (op, ind) = Xor(e, u);
    }
}
```

Figure 1: Using search time as a proof [1]

- Only two parties, the CSP and the client (generator and owner of the emails) A are considered.
- A sends emails to a number of recipients which are stored on a server.
- The communications are encrypted and thus only the ciphertexts are available on the server.
- A can be asked to present all emails containing a specific keyword. We ignore the presence of single or multiple recipients as we assume that all communication that must happen has happened and no more new emails will be generated (static setting).
- A is assumed to be honest- given a keyword, A queries the server using the same and returns all the emails that the server specifies as containing the given keyword.
- We utilise an index based SSE scheme. Consequently, there is an initial setup phase where A is required to generate the secure index and store it on the server. A is assumed to have one powerful device that can be used to create the server-side data structures the scheme requires. Once this done, A may send queries from a smaller, less powerful device that cannot store all plaintext emails but possesses the necessary secret information generated during the setup.
- Suppose A has numbered the emails 1, 2, 3, ... There is some third-party service that can send A the emails corresponding to index i. The bitmap based scheme under consideration does not return ciphertexts directly but only indicates the emails which

contain the keyword. This third-party service ensures that A can obtain the necessary emails without having to store all of them.

On the other hand the server

- stores all the data given to it honestly and does not modify or delete any of it
- may try to learn information about the underlying data from the ciphertexts
- may try to suppress some emails or claim some emails correspond to a keyword even though they do not

5 Implementation details

This sections presents the details of the algorithm used, the data used to test the implementation and some of the design decisions made to guarantee security.

5.1 Underlying algorithm

While several SSE schemes have been proposed, [2] presents the first dynamic and verifiable scheme. A verifiable scheme provides a mechanism to check that the returned documents indeed contain the keywords. A dynamic scheme allows documents and keywords to be added to and deleted from the server.

The first scheme proposed, which is our focus, is verifiable but not dynamic. This scheme includes the following algorithms:

- A randomized keygen algorithm that uses MACs to generate a integration of keys that will be applied in buildIndex. This is illustrated in Figure 2.

  KeyGen(κ) : This algorithm is run by the user to generate the set of keys used in the scheme. Choose three cryptographic MAC's defined as follows:

  – $H_1 : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^\kappa$
  – $H_2 : \{0,1\}^\kappa \times \{0,1\}^* \times \{0,1\}^\kappa \to \{0,1\}^\kappa$
  – $H_3 : \{0,1\}^\kappa \times \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^\kappa$

  Where, the first inputs are cryptographic keys. Output the key $\mathbf{K} = \langle K_1, K_2, K_e, K_h \rangle \xleftarrow{R} \{0,1\}^\kappa$

Figure 2: VSSE KeyGen [2]

- A randomized buildIndex algorithm that the client can run to generate a secure index. A secure index consists of a file table and a search table. The file table simply contains tuples of index and encrypted file. The search table consists of tuples of a keyword identifier, a bitmap, and a MAC tag. This algorithm takes as input some plaintext files and a key and outputs the index. This is shown in Figure 3.

  BuildIndex$(\mathbf{F}, \mathbf{K})$: This algorithm is run by the user to generate and output the secure index $\mathcal{I} = \langle T_f, T_s \rangle$. **Generation of T_f:** For $i = 1$ to m, compute $c_i = SKE.Enc(K_e, f_i)$ and store the tuple $\langle i, c_i \rangle$ as a row in T_f.
  Generation of T_s: Extract the keywords in $\mathbf{F}$ and set $\mathbf{W} = \{w_0, \ldots, w_n\}$ and for all w_i in $\mathbf{W}$:

  - Generate identifier $id(w_i) = H_1(K_1, w_i)$
  - Choose $r_i \xleftarrow{R} \{0,1\}^\kappa$ and obtain mask $h_i = H_2(K_2, w_i, r_i)$
  - Create the bitmap $B_m(w_i)[j] = 1$ for all $j \in$ index of encrypted files having the keyword w_i.
  - Compute $x_i = B_m(w_i) \oplus h_i$ and $Tag_i = H_3(K_h, B_m(w_i), id(w_i))$
  - Store $\langle key, value, Tag \rangle = \langle id(w_i), x_i || r_i, Tag_i \rangle$ as a row in T_s.

Figure 3: VSSE BuildIndex [2]

- A deterministic searchToken algorithm that the client can run to produce search tokens. This algorithm takes as input a keyword and a key and returns the keyword identifier. This is shown in Figure 4.

  SearchToken$(\mathbf{w}, \mathbf{K})$: This algorithm is executed by the user to generate search token.

  - Compute and output search token $\tau_s = \langle id(w) \rangle$

Figure 4: VSSE SearchToken [2]

- A deterministic search algorithm that the CSP can run with an input index and search token, and output obscured bitmap and verification tag. This algorithm scans the search table for the search token and returns the result if found. This is presented in Figure 5.

- A deterministic verify algorithm that the client can run with an input verification tag, obscured bitmap, keyword identifier, and key, and that outputs true or false. This algorithm recomputes the tag from the bitmap, identifier and key, and compares it to the input tag. This is presented in Figure 6.

Search$(\tau_s, \mathcal{I})$: This algorithm is executed by the CSP.

- Scan the search index $\mathcal{I}$ to locate $id(w_i)$.
- On success, return the tuple $\langle x_i||r_i, Tag_i\rangle$.

Figure 5: VSSE Search[2]

Verify$(\mathbf{x_i}||\mathbf{r_i}, \mathbf{Tag_i})$: This algorithm is run by the user to verify the correctness of the search outcome sent by CSP.

- Parse and extract r_i from $x_i||r_i$.
- Compute the keyword identifier $id(w_i) = H_1(K_1, w_i)$.
- Compute the mask $h_i = H_2(K_2, w_i, r_i)$.
- Extract the bitmap $B_m(w_i) = x_i \oplus h_i$.
- Calculate $Tag'_i = H_3(K_h, B_m(w_i), id(w_i))$.
- If $(Tag'_i = Tag_i)$ output $True$; else output $False$.

Figure 6: VSSE Verify[2]

For this work, we will require the email/file data along with keywords which can be accessed, encrypted and used by the algorithms. The collection of these is discussed subsequently.

5.2 Data

In the typical setting, it is usually the emails which characterize what the keywords will be i.e "keywords" are typically determined by the general inclinations of the people who write emails. For instance, if using only the subject line of an email as the source of the ciphertext to query on, one keyword could be "urgent" (which might indicate all matters that require immediate attention) or "Ameren" (indicating that a query has been made about matters regarding a particular organization) and results from the server can be returned correspondingly.

However, in this work we propose to discard such a notion. Instead we will first build up a dictionary of words which is substantially larger than the usual set of keywords used and compose files/emails using a random selection of these words. This decision is based on the reason that a FOIA query can ask for any keyword. In that case using a subset of the words used in previously written emails as a dictionary of keywords appears insufficient.

5.2.1 Keywords

To prepare a dictionary of words, we will use freely available text files of novels which can be processed and parsed suitably. We will likely obtain our text material from Project Gutenberg.

5.2.2 Emails

We will compose emails of different lengths using randomized permutations of the keywords. This will not use any information about the inherent structure of the English language.

5.3 Design decisions

[2] leaves several pieces of the SSE algorithms abstract, such as which symmetric key encryption algorithm to use, which cryptographic MACs are appropriate, and a value for the security parameter.

[2] states that in order for its claimed security properties to hold, the encryption scheme must be CPA secure. In order to satisfy this requirement, we used AES-CFB mode.

The work in [2] also requires that H_1, H_2, H_3 be secure MACs and in addition that H_3 must produce pseudorandom tags. We chose to use HMAC with SHA256 for all three MACs, as HMAC is a PRF assuming the compression function for the hash function is a PRF [10]. To the best of our knowledge, SHA256's compression function appears to be a PRF.

Finally, to choose the security parameter, we noticed that the keys in this scheme are only used in hashes and symmetric encryption. HMAC requires the key length to be at least the length of the hash output (256 bits in our case). A longer key does not provide much more security [11]. Since a key size of 256 bits is also compatible with AES, we chose to use 256 as the security parameter.

In terms of functionality, we follow the algorithms presented in [2] fairly faithfully. However, the algorithm described does not explain how to identify the actual emails given the server's response. In the FOIA setting, retrieving the emails is the most important part. Therefore, in our implementation, we used the bitmap extracted from the verify function to determine which emails contain the keywords.

6 Security Analysis

In this section, we discuss the security notion used for this work and some associated definitions.

6.1 IND-CKA2

The game for indistinguishability (of stored bitmap) against chosen keyword attacks can be described as follows-

- An adversary A chooses the set of emails F and generates the list of keywords W. It selects a $w^* \in W$ and shares F, W and w^* with challenger C.
- C generates the secure index I. During the generation of the index, when w^* is being handled, C selects a bit randomly from $\{0, 1\}$.
  - If b = 0, x^* is the encrypted version of the bitmap corresponding to w^*.
  - If b = 1, x^* is selected randomly from the range of the encrypted bitmap.
- C completes and returns the index I to A.
- A can generate search tokens for all $w \neq w^*$, call the verify function from Figure 6 and also ask for the decryption of bitmaps by supplying some w and corresponding $x||r$.
- The challenge consists of the keyword identifier $id(w^*)$.
- A outputs a bit b^*.

The scheme is said to be IND-CKA2 secure if for all PPT adversaries A,

$$P(b = b^*) \leq \frac{1}{2} + negl(\lambda)$$

where λ is the security parameter.

6.2 UF-CKA

Consider the following game

- An adversary A chooses the set of emails F and generates the list of keywords W. It selects a $w^* \in W$ and shares F, W and w^* with challenger C.
- C generates the secure index I. During the generation of the index, the tag for w^* is excluded.
- C completes and returns the index I to A.
- A can generate search tokens for all $w \neq w^*$, call the verify function from Figure 6 and also ask for the decryption of bitmaps by supplying some w and corresponding $x||r$. A can also query a tag generation oracle for w other than w^*.

- A outputs a forged tag Tag^* corresponding to the bitmap $B_m(w^*)$.

The scheme is UF-CKA secure if for all PPT adversaries A

$$P(Verify\_algorithm(Tag^*, x^*, id(w^*)) = Accept) \leq negl(\lambda)$$

where λ is the security parameter.

6.3 IND-CMA

Along with requiring the MAC schemes used in this work to have existential unforgeability, we will also require that the MAC tags generated by certain MACs be indistinguishable from elements selected randomly from the output (Tag) space of the MAC. A MAC scheme satisfying this is said to be IND-CMA secure.

6.4 VSSE

If the SKE scheme shown in Figure 3 is IND-CPA secure and H_2 is IND-CMA secure, then VSSE is IND-CKA2 secure. The IND-CMA security of H_2 guarantees that h_i is indistinguishable from randomly generated strings. Since x_i is obtained by the XOR operation with h_i, this essentially serves as a one time pad that ensures that the encrypted bitmaps are indistinguishable from randomly generated strings. The IND-CPA security of SKE ensures that the ciphertexts stored in T_f are indistinguishable from random.

Further, if the MAC H_3 has existential unforgeability, then VSSE is UF-CKA secure. This follows from the fact that the Tag_i generated in Figure 3 is based on H_3. If the probability of forging a MAC tag is negligible for the MAC scheme H_3, then the probability of forging a Tag that passes the verification test of Figure 6 is negligible. This in turn implies UF-CKA security.

7 Future work

Following a static setting, a natural extension would be to consider the dynamic setting for SSE. Here, keywords and encrypted files can be added or deleted from the server. This resembles the case of email ciphertexts stored on a server more than the static case does. A final step would be to attempt to build a GUI around the dynamic setting where users can send and receive email in an encrypted manner and still continue to search for emails containing a specific keyword.

While much of this work has focused on obtaining an email by querying a single word, a more practical application would be to obtain messages containing a specific phrase. However, for the model presented above, if there are n words in the dictionary and say the length of the phrase queried is at most k, then the size of the query space is n^k. Implementing the above scheme seems expensive as there is a dramatic increase in the number of keywords and keywords associated with the files.

A solution might lie in using the inherent structure of the English language- using a language model to identify a select phrases and building a dictionary on them. However, we believe this is beyond the scope of this work.

8 Conclusion

This work considers the problem of retrieving emails based on keyword queries from a cloud service that only possesses the ciphertexts of the emails under consideration. We restrict ourselves to the treatment of this problem in a static setting. This results in the implementation of a verifiable SSE scheme made available at https://github.com/Milind-Blaze/VSSE_and_DVSSE. We also identify valuable directions of future work that build on the work in [2].

References

[1] "Vfssse." https://github.com/zhangzhongjun/VFSSSE, 2020.

[2] R. Ramasamy, S. S. Vivek, P. George, and B. S. R. Kshatriya, "Dynamic verifiable encrypted keyword search using bitmap index and homomorphic mac," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 357–362, IEEE, 2017.

[3] "Freedom of information act (illinois)." https://research.illinois.edu/training/illinois-freedom-information-act-foia, 2021.

[4] "Freedom of information act (united states)." https://en.wikipedia.org/wiki/Freedom_of_Information_Act_(United_States), 2021.

[5] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *2012 IEEE International Conference on Communications (ICC)*, pp. 917–922, 2012.

[6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[7] "Clusion." https://github.com/encryptedsystems/Clusion, 2020.

[8] "awesome-sse." https://github.com/emad7105/awesome-sseimplementations.

[9] J. W. S. Y. H. Hwang and I. J. Kim, "Encrypted keyword search mechanism based on bitmap index for personal storage services," in *13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 140–147, 2014.

[10] M. Bellare, "New proofs for nmac and hmac: security without collision-resistance," in *Advances in Cryptology – CRYPTO '06*, pp. 140–147, 2014.

[11] R. C. H. Krawczyk, M. Bellare, "Hmac: Keyed-hashing for message authentication." https://datatracker.ietf.org/doc/html/rfc2104, 1997.