Volume XXIII 2020
ISSUE no.1
MBNA Publishing House Constanta 2020

Scientific Bulletin of Naval Academy

SBNA PAPER • OPEN ACCESS

Authentication protocol based on searchable encryption and multi-party computation with applicability for earth sciences

To cite this article: Marius Iulian Mihailescu, Stefania Loredana Nita and Ciprian Racuciu, Scientific Bulletin of Naval Academy, Vol. XXIII 2020, pg.221-230.

Available online at www.anmb.ro

ISSN: 2392-8956; ISSN-L: 1454-864X

doi: 10.21279/1454-864X-20-I1-030
SBNA© 2020. This work is licensed under the CC BY-NC-SA 4.0 License

Authentication Protocol based on Searchable Encryption and Multi-Party Computation with Applicability for Earth Sciences

Marius Iulian Mihailescu[1], Stefania Loredana Nita[2], Ciprian Racuciu[3]

[1]R&D Department, Dapyx Solution, Bucharest, Romania
[2]Computer Science Department, University of Bucharest, Romania
[3]Computer Science Department, University of Bucharest, Romania

E-mail: marius.mihailescu@hotmail.com

Abstract. The current paper will take the first stab in developing a high implementation of Multi-Party Protocol (MPC) using a searchable encryption scheme with steganographic structure hidden in multiple types of files. We formalize our results in a MpCSES (Multi-party Computation with Searchable Encryption Scheme), a tool-assisted framework for building a high-confidence cryptographic proof of our scheme. The case study that we have chosen to explain

1. Introduction

Cloud storage and file servers with Network Attached Storage (NAS) solutions are becoming increasingly popular and quite economically, being very attractive for users and companies who are dealing with large volumes of data. In a distributed system (e.g. cloud), the users are encrypting the data before uploading it to the cloud. In this case, the encryption process destroys the structure of the data and before processing the data needs to be decrypted. In order to repair this dilemma, searchable encryption is coming to help us [23], [24], [26]. Symmetric Searchable Encyrption (SSE) doesn't protect only the confidentiality of the data but also allows to search over encrypted data without being necessary the decryption of them. The schemes based on symmetric searchable encryption are more efficient compared with other cryptographic methods such as oblivious RAM (ORAM) which push the computational limits on the edge [20], [15].

A very important aspect is represented by the two new security notions which can be found entitled as forward and backward privacy. To handle the leakeages reminded previously, forward and backward privacy were firstly proposed in 2014 in the work [21].

For any adversary that is trying to observe the interaction between the server and the client, the forward privacy will be satisfied if an addition of new files will not have any kind of leakage of any information about previous query keywords. Backward privacy will occur if the files that were added previously and later they were deleted doesn't leak excessively knowledge into an amount period of time in which two search queries took place based on the same keyword. Bost [16] proposed forward privacy and created a DSSE system that resits to attacks based on injectiong files [17]. Later, Zuo et al. [13] improved the system in order to support queries with ranges.

Bost et al. [14] defined recently in one of his workpapers three types of backward privacy, starting with a stronger form up to a weaker form, as follows:

- *Type-I* - Backward privacy based on the pattern for file inserting. Let us take into consideration a keyword *w* and the amount of time passing between two consecutive searches of the keyword *w*. The type-I of backward privacy reveals knowledges about the time of inserting new files which include *w* and the number of update operations.
- *Type-II* - Backward privacy based on the pattern for file updating. This type will leaks when all updates and also deletion related to w will take place.
- *Type-III* - Weak backward privacy. This type will reveal knowledge in the moment in which a previous operation of adding a file was repealed by a specific operation of file deleting.

Paper structure. The paper is structured as follows:

- *Section 1 - Introduction*. The section gives a brief introduction in the field of Multi-Party Computation and Searchable Encryption Schemes, helping us to see exactly were we can place the current research and which are the benefits that can be gained.
- *Section 2 - Related work*. The objective of the section is to create a guide of all the notations in order to help the reader to follow in a easy and practical manner the current research paper.
- *Section 3 - The proposed scheme*. A brief start-of-the art of the most important researches from 2016 to 2019 are presented, pointing out the advantages of the research conducted by the authors. The section is focusing on Searchable Encryption and Multi-Party Computation Protocols.
- *Section 4 - Conclusions and Future Research Directions.* The section will describe briefly the scheme and its components. Security and correctness proofs are discussed. Each step of the schemes is deeply analysed by giving all the necessary elements to understand theoretical side and easy to follow if the scheme will be deployed by the reader in a real environment.

Our contributions. Our paper will introduce a Multi-party protocol based on searchable encryption, with a trapdoor adjusted to take and work over steganographic objects. The main goals proposed in this paper are:

1. Defining a *multi-party protocol* based on searchable encryption mechanisms;
2. Defining a searchable encryption scheme based on security parameter from the multi-party protocol and steganographic objects with respect for:
    - Defining a *trapdoor function* and adjusting it to receive steganographic objects with encrypted values and to operate with them;
    - Defining a *search function* that is able to search over a trapdoor using the security parameter λ;
    - Discussing and demonstrate the proof of correctness and proof of security.
3. Designing and implementing an algorithm for generating *user identifier* based on user name, personal unique identifier, and date of birth. The output of the algorithm will be a user identifier which is based on the three elements mentioned above.

2. Related work

2.1. Searchable Encryption.

Searchable encryption consist in the process of searching over encrypted data that are located on a trusted/untrusted server or cloud without being necessary to decrypt the data [22].

In 2000 Song et al. proposed an idea which solved the issue of searching encrypted data. Bosch et al [18] give a very intersting examination of the techniques used by searchable encryption, techniques such as: Searchable Symmetric Encryption (SSE), Public Key with Keyword Search (PEKS), Identity-Based Encryption (IBE), Hidden Vector Encryption (HVE), Predicate Encryption (PE), and Inner Product Encryption (IPE). Later, recent study [19] showed that Multi-keyword Rank Searchable Encryption (MRSE) is a new searchable encryption key.

Recently studies focused on Dynamic Searchable Encryption (DSE), such as [1], [2]. In [1], Zuo et al. discuss about Dynamic Searchable Symmetric Encryption (DSSE) based on a simple symmetric encryption that contains the addition operation with homomorphic properties and bitmap index

method. Blomer and Loken [2] introduce a dynamic searchable encryption scheme based on access control focusing on features such as fine-grained access control for searching the results, as well on controlling the access on operations such as adding documents to the document collection. The construction proposed by Bakas [3] supports a multi-client model, in the way that every user will hold a secret key used for performing search queries. Wang et al. [4] propose a very interesting approach based on encrypted index which will allows them to have queries for a sequence of keywords, by introducing an update for DSSE to support range query.

Starting with 2018, we have seen a trend on adopting Attribute-Based Encryption within a Searchable Encryption scheme with the goal of having a flexible data sharing. A first step in this direction represents the effort of Antonis Michalas in [8]. In [9], Lai et al., proposed HVE scheme which utilizes effective symmetric-key building blocks in order to elimitate completely operations that involve the elliptic curves.

Guaranteeing security in searchable encryption schemes represents a critical aspect and it need to taken into consideration when such schemes are designed, especially on designing trapdoors functions. In [10], the authors introduces a novel inference attack targeting the traditional databases (namely, relational ones) that are secured via SSE schemes. The attack proposed lets a passive adversary to retrieve the names of the attribute included in certain detected queries. Note that the adversary learns just a few characteristics about the information of the meta-data.

2.2. Multi-Party Computation Protocols

Multiparty Computation enables n untrutfull parties to compute on an agreed upon function using some private inputs x_i.

In [5] the authors are taking the first step in developing high-assurance implementation in Proactive Multi-Party Computation - P(MPC). Shlomovits et al. [6] propose a ShareLock framework that represents a concrete tool for privacy increasing for cryptographic currencies and it can be deployed on nowadays specific networks for cryptocurrency. In [7] the authors propose a secure-MPC protocol in plain model composed out of four round which achieves security against any dishonest majority. The existence of four rounds specific to oblivious transfer represents the security of the protocol.

A very interesting approach of MPC is the one with fully homomorphic encryption. An important step to this has been done in 2018 in [11] in which Li et al. propose a decryption process (providing a very good example of distributed decryption) that can be implemented for each user that is participating in an independent manner, and also reducing the number of interactions that characterize the users during a communication process and that are finding themselves in the decryption process.

In [12], the authors are introducing for the first time a truly practical full threshold ECDS signing protocol which is based on both, fast signinging and fast key distribution. With this, the authors are resolving an important problem, opened for many years. With their results obtaiened, we can see that a series of applied uses of threshold ECDSA signature are opened and they are very demanding nowadays.

Our current paper, presents the first attemptive of a MPC protocol with searchable encryption capabilities, trying to solve an old open problem.

3. The Proposed Scheme

Our scheme (see Figure 1) is composed from three components: multi-party protocol, searchable encryption scheme, and trapdoor and steganography scheme. The files are stored on the server side and each file (text, audio, video) has an steganographic object hidden, see Section 3.3.

Component 1 – Multi-Party Protocol

The role of the multi-party protocol (see Section 3.1) is to make sure through a trusted party that all the parties (including the server side) doesn't have access directly on the data that is exchanged between the parties and the server. Each party will store a private value x_i and all the parties will

compute an agreed and defined function. When the protocol will end its execution he will output a security parameter θ which will be passed, further to the Searchable Encryption Scheme.

Component 2 – Searchable Encryption Scheme
The uniqueness of the searchable encryption scheme consist in Step 4 – $GeneratingTrapdoor(secret_{key}, keyword)$ and Step 5 - $Search(Trapdoor_{keyword}, \lambda)$. In Step 4, based on the secret key and keyword as parameters, the trapdoor is generated in such way that based on the secret key of an user, his files will be returned and the search can continue for a specific keyword.

Component 3 – Steganography Scheme
We started from the idea that a user has multiple types of files (text, audio, video). How a searchable encryption scheme will look like? For this we designed a steganographic object (see Section 3.3) which consist from two main elements, the user identifier and the keywords associated with that file. The extra ingredient on this elements is that they are encrypted before they are hidden in the files and once they are extracted they need to be decrypted. The encryption of the elements are done with AES and RSA algorithms.

Component 2 and 3 are included in the TP of the Multi-Party Protocol (see Figure 2).

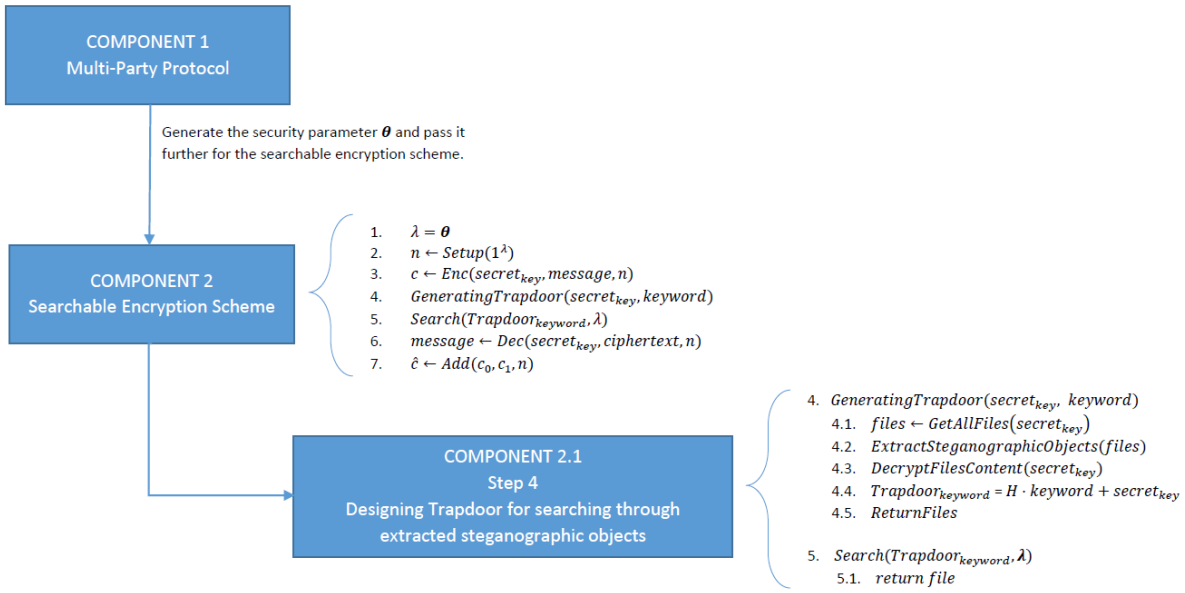

Figure 1. Components overview

3.1. Multi-Party Protocol

Let's assume that we have m parties. Each party P_i will hold a private value x_i and all the parties they will want to compute some agreed function:

$$sessmpc(x_1, x_2, \dots, x_m) = x_1 \cdot X^{m-1} + 2 \cdot x_2 \cdot X^{m-2} + \cdots + m \cdot X_m \cdot X^{m-m} \quad (1)$$

For our case, we will start with a protocol which enables alike computation, that is correct. By first correctness it is clear: the calculated outcome resulted from the protocol is doubtlessly the correct value of $sessmpc(x_1, x_2, \dots, x_m)$.

The ideal functionality for computing function sessmpc:

$$sessmpc: (\{0,1\}^* \times \{0,1\}^* \times R) \rightarrow (\{0,1\}^* \times \{0,1\}^*) \quad (2)$$

where R represents the domain of the random inputs.

The protocol that we use for our scheme is the following:

1. get the input x_0 from P_0 and x_1 from P_1.
2. get the corrupt i message from the S, $i \in \{2, \dots, m\}$
3. give x_i to S.
4. get x_i' from S.
5. computes $(y_0, y_1) = f(x_1^{"}, x_2^{"})$ where $x_i^{"} = x_i'$ and $x_{i-1}^{"} = x_{i-1}$.
6. give y_i to S.
7. get y_i' from S.
8. when S says "move j" give $y_j^{"}$ to P_j where $y_j^{"}$ can be y_i' if $i = j$ or y_i if $i \neq j$.

Further, we will continue by defining the security.

Definition 1 (Probability Ensemble). A probability ensemble represents an infinite sequence of distributions $\Gamma = \{D_n\}_{n\in\mathbb{N}}$ where each of D_n represents a probability distribution. We will think about D_n as describing an experiment that has a security parameter dependent on n.

Definition 2. The protocol, which is statistical indistinguishably, π will securely realize TP if $\forall PPT$ A $\exists$PPT S $\forall$ PPT E the ensembles

$$\{Exec_{F,S,E,n,z}\}_{n\in\mathbb{N},z\in\{0,1\}^{polynom(n)}} \quad (3)$$

are indistinguishable, where $\{Exec_{\pi,A,En,z}\}$ represents the output of the Observer in the ideal model.

In our model $\theta \in \mathbb{N}$ represents the security parameter which will be used as well in the searchable encryption scheme and $z \in \{0,1\}^{polynom(n)}$ represents the Observer input.

Our ideal model can be generalized to the case of multiparty computation. If we will allow the "corruption" message to take place in the beginning of the protocol running, such protocol will be called Static Corruption Model (SCM). If the "corruption" message will take place in each round of the model, then we have a Dynamic Corruption Model (DCM).

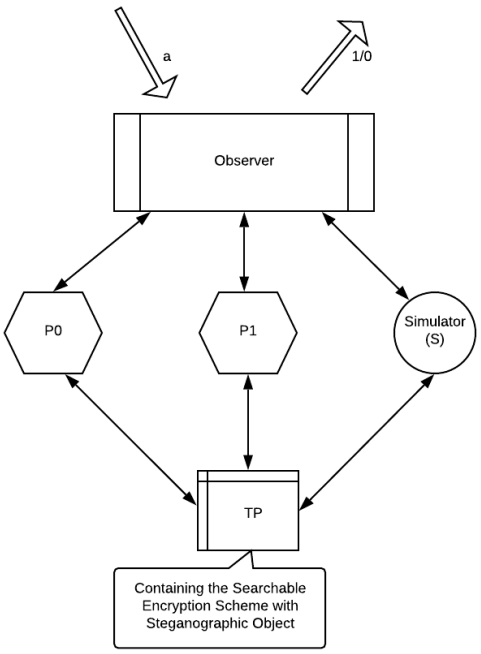

Figure 2. The Ideal Model

3.2. Multi-Party Protocol

Our searchable encryption scheme is a simple symmetric encryption that contains the addition operation with homomorphic properties Ω. Such scheme consist of four algorithms: *Setup*, *Enc*, *Dec*, and *Add*. Each step is described bellow. In our searchable encryption scheme λ is θ from **Definition 2**.

1. $n \leftarrow Setup(1^{\lambda})$: Based on the security parameter, the *Setup* algorithm will output a public parameter n. The parameter will have the following value $n = 2^{l}$ where l represent the maximum number of files that the scheme is able to support.
2. $c \leftarrow Enc(secret_{key}, message, n)$: The *Enc* algorithm will have as parameters: the message m, a public parameter n, and a random secret key ($secret_{key}$), $0 \leq sk < n$. Based on the parameters we will compute the ciphertext as $ciphertext = secret_{key} + message \mod n$, where *message* is $0 \leq message < n$. We have to remember that for every encryption, the $secret_{key}$ needs to be stored, and it will be used only once.
3. $GeneratingTrapdoor(secret_{key}, keyword)$: one of the most important component of our searchable encryption scheme consist in generating the trapdoor and guaranteeing its security. The user, who is the owner of the file(s) will generate the trapdoor according to the *keyword*, which is searched using the $secret_{key}$. The trapdoor $Trapdoor_{keyword}$ is sent to the user. The trapdoor itself is dependent on the secret key. In this case we have eliminated the possibility of online guessing the keyword. The trapdoor is defined as follows: $Trapdoor_{keyword} = H \cdot keyword + secret_{key}$, where $H: \{0,1\}^{*} \rightarrow \{0,1\}^{*}$.
4. $Search(Trapdoor_{keyword}, \lambda)$: The scheme give the possibility for searchable encryption to use a trapdoor. To search a keyword, the user send the keyword to the server. On the server, the steganographic objects are extracted from the files and the values from the files are decrypted based on the λ. The files that mach the user identifier and keyword(s) are than returned to the user.
5. $message \leftarrow Dec(secret_{key}, ciphertext, n)$: For the *ciphertext* obtained, the public parameter n and the $secret_{key}$, the message will be recovered by computing as $message = c - secret_{key} \mod n$.
6. $\hat{c} \leftarrow Add(c_0, c_1, n)$: We assume that we have two ciphertexts, c_0 and c_1, and the public parameter n, the algorithm will compute $\hat{c} = c_0 + c_1 \mod n$, where $c_0 \leftarrow Enc(secret_{key_0}, m_0, n) \leftarrow Enc(secret_{key_1}, m_1, n)$, $n \leftarrow Setup(1^{\lambda})$ and $0 \leq secret_{key_0}, secret_{key_1} < n$.

One interesting challenging it will be obtaining a system recommendation and analytics in a big data environment [27]. This can be done in future as a research direction by extending the searchable encryption scheme and the MPC protocol by integrating also support for searching analytics and making recommendations based on them.

3.2.1 Proof of security [25]

We will state that Ω is (perfectly) secure if ∀ probabilistic polynomial time (PPT) adversary, A, the advantage on their security game is negligible or if $n \leftarrow Setup(1^{\lambda})$, the secret key ($0 \leq secret_{key} < n$) is kept secret and A will choose m_0, m_1 where $0 \leq m_0$ and $m_1 < n$.

$$Adv_{\Omega,A}^{PS}(\lambda) = |Pr[A(Enc(secret_{key}, message_0, n)) = 1] - Pr[A(Enc(secret_{key}, message_1, n))]| \leq \epsilon \quad (4)$$

Theorem 1. The scheme proposed in this article is perfectly secure.

3.2.2 *Proof of correctness*

For our scheme, the correctness consist in the sum of the two ciphertexts $\hat{c} = c_0 + c_1 \mod n$ which will decrypts to $message_0 + message_1 \mod n$ using the $secret_{key} = secret_{key_0} + secret_{key_1} \mod n$. This can be said as following as well:

$$Dec(\hat{s}ecret_{key}, \hat{c}, n) = \hat{c} - \hat{s}ecret_{key} \mod n = message_0 + message_1 \mod n \quad (5)$$

As being said, this requirement is very easy to be checked.

We will choose the plaintext space M, keystream space K, assuming that $\mathcal{K} = |M|$, $m \in [0, M-1]$, $c \in [0, M-1]$. Set $k^* = c - m(modM)$. Then:
$Prob_k \leftarrow \mathcal{K}[Enc(k,m,M) = c] = Prob_k \leftarrow \mathrm{K}[k + m = c(modM)] = Prob_k \leftarrow K [k = c - m(modM)] = Prob_{k \leftarrow K}[k = k^*]$.

For encryption and decryption algorithm of Ω, the secret key is used only once.

3.3. *Steganographic object*

The steganographic object (text file) that will be hidden in each of the user files contains two important elements: the encrypted values of the user identifier and of the keywords. The user identifier is generated as described in Section 3.4. The encryption of the values we have done it using RSA and/or AES algorithms. The $privatekey$ is the $secret_{key}$ from step 4.1 (see Figure 1).

The encrypted values will be stored as follows:

$$Enc_{private_{key}}(UID) \ \# \ Enc_{private_{key}}(keyword_1) \ \% \ \ldots \ \% \ Enc_{private_{key}}(keyword_n) \quad (6)$$

In Equation 6 we can observe two special characters, # and %. The special characters are used as best practices and to separate the values between them and also they are very useful during the implementation of the algorithm.

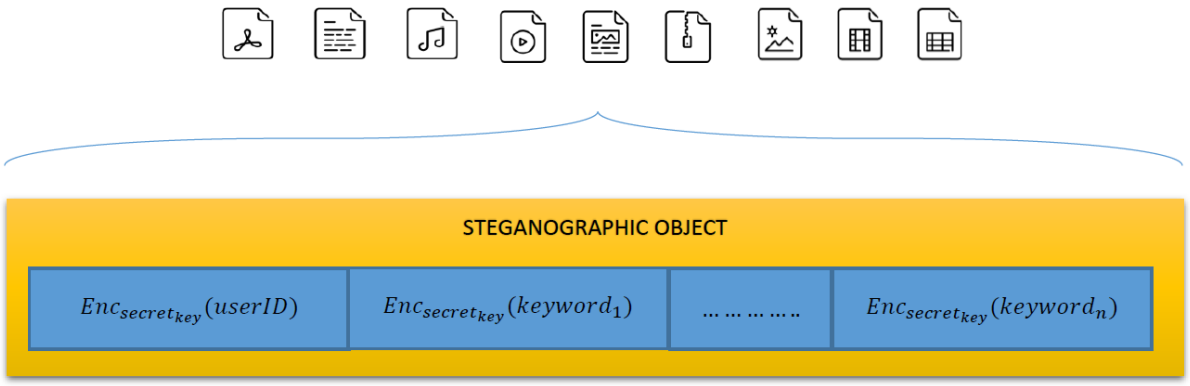

Figure 3. Steganographic Object

3.3. *Generating user unique identifier algorithm*

The algorithm described in this section will help to set for each file its unique identifier. The unique identifier is designed and based on user name, personal unique identifier, and date of birth.

The below algorithm describes a different way of generating an identifier for the users. We choose a different approach compared to hash functions in order to avoid the difficulties that can be raised

during the implementation process. This algorithm has been designed for experimental purpose. For specific string processing software applications, for example spell-checking, the hash table are less efficient than tries, finite automata or other array concepts (e.g. Judy arrays).

Each input parameter (user name, personal unique identifier, and date of birth) are represented by a small number of bits compared to a hash table, one may use the input directly as the index for an array of values. In this case we will have no collisions.

Algorithm 1. User Unique Identifier Generating Algorithm

```
1.   input: User Name (U_N), personalUniqueIdentifier(PU_ID), dateOfBirth(d_B)
2.   output : uniqueIdentifier(u_ID) for a given user

3.   method GenerateID(string userName, string UniqueID, Date dateObject)
4.          format dateOfBirth(d_B) as "mmddyyyy"
5.          declare u_id
6.          u_id ← empty string
7.          while u_id IS empty OR u_id exist do
8.              declare randomN
9.              randomN ← GenerateRandomNumber(0, 3000000)
10.             declare startEdge and endEdge
11.             startEdge ← randomN mod userName.length()
12.             endEdge ← (randomN + 4) mod userName.length()

13.             if endEdge > startEdge then
14.                 userName ← userName.substring(startEdge, endEdge)
15.             else
16.                 userName ← userName.substring(startEdge, userName.length()
                                                  −1)+userName.substring(0, endEdge)
17.             end if

18.             startEdge ← randomN mod UniqueID.length()
19.             endEdge ← (randomN + 4) mod UniqueID.length()

20.             if endEdge > startEdge then
21.                 UniqueID ← UniqueID.substring(startEdge, endEdge)
22.             else
23.                 UniqueID ← UniqueID.substring(startEdge, userName.length()
                                                  −1)+UniqueID.substring(0, endEdge)
24.             end if

25.             startEdge ← randomN mod dateObject.length()
26.             endEdge ← (randomN + 2) mod dateObject.length()

27.             if endEdge > startEdge then
28.                 dateObject ← dateObject.substring(startEdge, endEdge)
29.             else
30.                 dateObject ← dateObject.substring(startEdge, userName.length()
                                                  −1)+dateObject.substring(0, endEdge)
31.             end if

32.             declare stringRow
33.             declare reversedString
34.             declare UID
```

Figure 4. The pseudocode for generating user unique identifier

4. Conclusions

In this paper, we propose a multi-party protocol using searchable encryption and steganographic object. All goals proposed in Section 1 they have been fulfilled with success. Currently, our scheme supports single keyword queries.

For future research directions we have in mind the followings:

1. Implementation of the full multi-party protocol in a real environment;
2. Performing security attacks and games on the protocol;
3. Analysis backward and forward privacy and examines the leakages;
4. Improving the steganographic structure and adding support for more encryption algorithms.

References

[1] Zuo C., Sun SF., Liu J.K., Shao J., Pieprzyk J. (2019) Dynamic Searchable Symmetric Encryption with Forward and Stronger Backward Privacy. In: Sako K., Schneider S., Ryan P. (eds) Computer Security ESORICS 2019. ESORICS 2019. Lecture Notes in Computer Science, vol 11736. Springer.

[2] Blömer, Johannes and Nils Löken. Dynamic Searchable Encryption with Access Control. IACR Cryptology ePrint Archive 2019 (2019): 1038.

[3] Bakas, Alexandros and Antonis Michalas. Multi-Client Symmetric Searchable Encryption with Forward Privacy. IACR Cryptology ePrint Archive 2019 (2019): 813.

[4] Wang, Jiafan and Sherman S. M. Chow. Forward and Backward-Secure Range-Searchable Symmetric Encryption. IACR Cryptology ePrint Archive 2019 (2019): 497.

[5] Eldefrawy, Karim and Vitor Pereira. A High-Assurance Evaluator for Machine-Checked Secure Multiparty Computation. (2019).

[6] Shlomovits, Omer and István András Seres. ShareLock: Mixing for Cryptocurrencies from Multiparty ECDSA.IACR Cryptology ePrint Archive 2019 (2019): 563.

[7] Choudhuri, Arka Rai, Michele Ciampi and Vipul Goyal. On Round Optimal Secure Multiparty Computation from Minimal Assumptions. IACR Cryptology ePrint Archive 2019 (2019): 216.

[8] Michalas, Antonis. The lord of the shares: combining attribute-based encryption and searchable encryption for flexible data sharing. SAC (2018).

[9] Lai, Shangqi, Sikhar Patranabis, Amin Sakzad, Joseph K. Liu, Debdeep Mukhopadhyay, Ron Steinfeld, Shifeng Sun, Dongxi Liu and Cong Zuo. Result Pattern Hiding Searchable Encryption for Conjunctive Queries. IACR Cryptology ePrint Archive (2018).

[10] Abdelraheem, Mohamed Ahmed, Tobias Andersson, Christian Gehrmann and Cornelius Glackin. Practical Attacks on Relational Databases Protected via Searchable Encryption. ISC (2018).

[11] Li, Ningbo, Tanping Zhou, Xiaoyuan Yang, Yiliang Han, Longfei Liu and Wenchao Liu. Two round multiparty computation via Multi-key fully homomorphic encryption with faster homomorphic evaluations. IACR Cryptology ePrint Archive 2018 (2018): 1249.

[12] Lindell, Yehuda and Ariel Nof. Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody.ACM Conference on Computer and Communications Security (2018).

[13] Zuo, C., Sun, S.F., Liu, J.K., Shao, J., Pieprzyk, J.: Dynamic searchable symmetricencryption schemes supporting range queries with forward (and backward) security.In: ESORICS 2018. pp. 228-246. Springer (2018).

[14] Bost, R., Minaud, B., Ohrimenko, O.: Forward and backward private searchableencryption from constrained cryptographic primitives. In: CCS 2017. pp. 1465-1482. ACM (2017).

[15] Garg, S., Mohassel, P., Papamanthou, C.: Tworam: Efficient oblivious ram in tworounds with applications to searchable encryption. In: Annual Cryptology Confer-ence. pp. 563-592. Springer (2016).

[16] Bost, R.:Forward secure searchable encryption. In: CCS 2016. pp. 1143-1154. ACM (2016).

[17] Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: The powerof file-injection attacks on searchable encryption. In: USENIX Security Symposium.pp. 707-720 (2016).

[18] C. Bosch, P. Hartel, W. Jonker, and A. Peter, A Survey of Provably Secure Searchable Encryption, ACM Comput. Surv., vol. 47, no. 2, pp. 1-51, 2014.

[19] R. Li, Z. Xu, W. Kang, K. C. Yow, and C. Z. Xu, Efficient multi-keyword ranked query over encrypted data in cloud computing, Futur. Gener. Comput. Syst., vol. 30, no. 1, pp. 179-190, 2014.
[20] Wang, X.S., Nayak, K., Liu, C., Chan, T., Shi, E., Stefanov, E., Huang, Y.: Oblivious data structures. In: CCS 2014. pp. 215-226. ACM (2014).
[21] Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryptionwith small leakage. In: NDSS 2014. vol. 71, pp. 72-75 (2014).
[22] J. L. Fernandez-Aleman, I. C. Senor, P. Angel O. Lozoya, and A. Toval, Security and privacy in electronic health records: A systematic literature review, J. Biomed. Inform., vol. 46, no. 3, pp. 541-562, 2013.
[23] Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Ro Su, M.C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In:CRYPTO 2013, pp. 353-373. Springer (2013).
[24] Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryp-tion: improved definitions and efficient constructions. In: CCS 2006. pp. 79-88.ACM (2006).
[25] Castelluccia, C., Mykletun, E., Tsudik, G.: Efficient aggregation of encrypted data in wireless sensor networks. 3rd intl. In: Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Sensor Networks, Italy (2005).
[26] Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: S P 2000. pp. 44-55. IEEE (2000).
[27] N. S. Loredana, "On recommendation systems applied in big data," *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Ploiesti, 2016, pp. 1-6.