LocLess: Do You Really Care Where Your Cloud Files Are?

Michalas, A. and Yigzaw, K.Y.

LocLess: Do you Really Care Where Your Cloud Files Are?

Antonis Michalas
Cyber Security Group,
Department of Computer Science
University of Westminster,
London, UK
a.michalas@westminster.ac.uk

Kassaye Yitbarek Yigzaw
Department of Computer Science,
UiT The Arctic University of Norway,
Norwegian Centre for E-health Research,
Tromsø, Norway
kassaye.y.yigzaw@uit.no

***Abstract*—Physical location of data in cloud storage is a problem that gains a lot of attention not only from the actual cloud providers but also from the end users' who lately raise many concerns regarding the privacy of their data. It is a common practice that cloud service providers create replicate users' data across multiple physical locations. However, moving data in different countries means that basically the access rights are transferred based on the local laws of the corresponding country. In other words, when a cloud service provider stores users' data in a different country then the transferred data is subject to the data protection laws of the country where the servers are located. In this paper, we propose LocLess, a protocol which is based on a symmetric searchable encryption scheme for protecting users' data from unauthorized access even if the data is transferred to different locations. The idea behind LocLess is that *"Once data is placed on the cloud in an unencrypted form or encrypted with a key that is known to the cloud service provider, data privacy becomes an illusion"*. Hence, the proposed solution is solely based on encrypting data with a key that is only known to the data owner.**

***Index Terms*—Security, Cloud Computing, Data Protection, Storage Protection, Searchable Encryption, Location Sensitive**

I. Introduction

While location-based services are becoming more and more popular we have now reached to a point where even novice users are concerned about protecting their privacy. More precisely, many users believe that giving away private information, such as their exact geographical location, age, home-town etc., lead to blatant violation of their privacy. As a result, they try to protect their privacy by evaluating all services before they actually start using them. One of the first things that users try to protect is their actual location. To this end, they try to avoid being tracked by simply turning off the corresponding location services from the applications that have installed on their smart-phone devices.

Even though keeping the location of individuals private is becoming more and more popular and companies try to build mechanisms that will eventually provide the necessary guarantees to the users, things seems to be a bit different when it comes to cloud-based services [1], [2]. Lately, along with the already traditional questions about the safety of cloud environments we have also seen concerns about the physical location of data. More precisely, users demand to be able to track the physical location of the data that they store on the cloud. In other words, users take the role of a spying agent who tries to breach the privacy of the the cloud service provider. This problem is known as the geolocation of data placement in cloud environments.

When users store their data on a cloud service provider, they hand it over to a provider that may have data centres in different geographical locations, countries or even continents. Moreover, it is a common practice that cloud service providers create replicas of the received data in order to protect users' information from possible failures of the initial servers that are storing the data. As a result, users' data are transferred between multiple servers that might be located in several different locations around the world, and they might be even managed and processed by external data centres.

The generation of data replicas and the placement of data in multiple remote locations has as a result that user' may never know where exactly her data and the corresponding replicas are stored. While this seems to be an innocent practice and in many cases would not affect the experience of the end user', it is considered as an important security flaw that can lead to the breach of privacy of cloud users'. The main reason for this, is the fact that if the cloud service provider has servers in a foreign country, the laws of that country may govern users' data when stored in that server. As a result, many important foreign laws may govern users' data and a third party can request access to the data of a certain user' directly from the cloud service provider. In such a case, user's data may be revealed without her consent, or even her knowledge.

Cloud service providers want the freedom to move data to different servers for load balancing or to take advantage of the lower cost of utilities or personnel in different geographies. However, by doing so, they may inadvertently expose their customers' data to the laws of countries other than those where the customer opted to operate. Hence, while a cloud service provider may take advantage of the friendly business environment in a country, it may also subject equipment and data stored in this equipment to the monitoring and surveillance of the government in that country.

There is a significant number of proposed protocols [3]–[9] that try to offer a reliable solution with which users will be able to verify at any time the exact geographical location of her data as well as the corresponding replicas. However, it has been observed that these approaches have many limitations. In addition to that, some of the existing solutions rely on unrealistic assumptions. For example, the existence of a GPS

device in the server rooms. However, server rooms are usually located in isolated places where there is no GPS signal. Hence, a solution based on such an assumption is considered as quixotic.

A. Our Contribution

In this paper, we present LocLess – a protocol through which cloud users' will be able to store their data on the cloud in such a way that location of the underlying servers will not affect their privacy. More precisely, after analyzing the existing geolocation systems for the cloud, we make the following argument:

"Once data is placed on the cloud in an unencrypted form or encrypted with a key that is known to the cloud service provider, data privacy becomes an illusion".

Driven by this argument, we propose a protocol that allows cloud users' to upload data of two different types:

1) Data that should always remain private. In other words, even if the cloud service provider transfers the data or replicas of the data in a location where a third party might be able to access no valuable information about the actual content will be revealed.
2) Data that users' do not necessarily wish to protect from unauthorized access. Meaning that the actual location of the underlying server is irrelevant.

To achieve the first goal, we propose a protocol that is solely based on a Symmetric Searchable Encryption scheme [10] while for the second one we rely on standard techniques for secure upload of data on the cloud.

B. Organization

In Section II, we present the state-of-the-art research for verifying the physical location of data in cloud storage. In Section III, we define the problem of trusted geolocation in cloud computing and the primitives used throughout the paper, while in Section IV we describe the proposed protocol. In Section V, we provide a security analysis and in Section VI we conclude the paper.

II. Related Work

In this section, we present the most important works that have been proposed in order to tackle the problem of data geolocation in cloud environments. In addition to that, we present a set of cryptographic tools that have been used by several solutions.

*Proof of Retrievability (*PoR*):* PoR was introduced in [11] and is a cryptographic proof of knowledge scheme which enables a user (verifier) to determine whether a host (prover) possesses a file f. More precisely, a host can prove to a user that she can retrieve the file without having knowledge of f. A PoR scheme consists of five algorithms, but for simplicity, in the rest of the paper we denote by $\mathsf{POR}(\mathsf{P}, \mathsf{V})$ an execution of the protocol between a prover P and a verifier V. PoR schemes have used by many protocols in order to provide a reliable solution to the problem of physical location of data in cloud storage.

Watson *et al.* [3] argued that there are limitations to the accuracy of verifying the location of data in a cloud storage. Authors showed that when a corrupted cloud service provider colludes with malicious hosts, it is infeasible for a user' to correctly verify the exact geographical location of the stored data. Moreover, authors were the first to take into consideration cases where two or more malicious hosts collude and create replicas of the stored files. This assumption led them to argue that the task of restricting where the geographic location of data is *impossible*. Additionally, they proposed a proof of location (PoL) scheme that can be used by a user in order to obtain the location of a stored file.

Benson *et al.* [6] proposed a protocol for approximately calculating the location of data in Infrastructure-as-a-Service storage with a per-data center granularity. The solution assumes that the locations of all data centres where the cloud service provider stores data are known, that the cloud service provider does not have any exclusive Internet connection between the data centres and that for each data center, there is a trusted third party node located geographically close to it, relative to the distance between the data centres. The proposed method relied on the Haversine distance as a passive distance measurement between the data centers to determine the location of the data centres where a certain piece of data is stored. Moreover, the paper discusses techniques to determine the location without having the list of data centres disclosed and detect the changes within a location. Apart from the proposed method itself, the authors contribute with a solid overview of the cloud data geolocation approaches.

Albeshri *et al.* [4], [5] proposed a protocol which combines a PoR scheme with a time-based distance-bounding protocol to determine the distance between a data centre and a verifier. The proposed solution assumes that a tamper-proof GPS device is attached to the local network of the cloud service provider and a third party will communicate with this device in order to verify the location of the stored data on behalf of a user'.

Gondree and Peterson [7] proposed a Constraints-Based Data geolocation (CBDG) solution for determining the location of data and its "binding" to specific locations. More precisely, authors extended the solutions proposed in [6], [8] by designing a generic framework for actively monitoring the location of stored data in the cloud using latency based techniques. The suggested approach combined probabilistic provable data possession with geolocation in a CBDG protocol, which is solely based on a PoR scheme.

One of the most promising solutions is the one presented in [12] where authors presented SecLoc. SecLoc is based on Key-Policy Attribute-Based Encryption [13] and achieves data confidentiality, location-sensitivity and computing efficiency. More precisely, SecLoc ensures that the cloud user's data is stored and can be processed only at locations that satisfy user specified location constraints. If the cloud providers (unintentionally) copy the user data to nodes outside the expected regions, the user data will become inaccessible at those ineligible locations.

III. Problem Statement & Definitions

In this section, we define the problem of geolocation in cloud computing along with the primitives that we use in the rest of

the paper. Furthermore, we explicitly define the capabilities of the adversary by defining the threat model that we consider.

Cloud Service Provider (CSP): We consider a cloud computing environment based on a trusted IaaS provider like the one described in [14] and [15]. The IaaS platform consists of cloud hosts which operate virtual machine guests and communicate through a network. In addition to that, we assume a PaaS provider is built on top of the IaaS platform and can host multiple outsourced databases. Furthermore, the PaaS provider offers an API through which a developer can built a privacy-privacy preserving application that offers searchable encryption functionality like the one presented in [16] and [17].

CSP's Locations & Hosts: We assume that a CSP uses a set of geographically distributed hosts. Let $\mathcal{L} = \{l_1, \ldots, l_n\}$ be the set of all locations where CSP can store data. Then the set $\mathcal{S}_i = \left\{s_1^i, \ldots, s_k^i\right\}$ is defined as the set of all hosts owned by CSP in a location l_i.

Trusted User Locations: Each user u_i who wishes to store a file f needs to define a list of trusted locations. Let $\mathcal{T}_i \subseteq \mathcal{L}$ be the set of all trusted locations for user u_i. Then the set of hosts in $\mathcal{T}_i$ is denoted as $\mathcal{S}_{T_i} = \left\{s_1^{T_i}, \ldots, s_l^{T_i}\right\}$.

Distance Between a Host & a Location: Most of the protocols that are dealing with the problem of physical location of data in cloud storage are using distance bounding techniques in order to measure the distance between a host and a location. Therefore, we denote the distance between a host s_k^i and a location l_j as follows:

$$\text{dist}\left(s_k^i, l_j\right) = \begin{cases} 0, & \text{if } s_k^i \text{ is located in } l_j \\ |l_i - l_j|, & \text{otherwise} \end{cases}$$

One of the core components of our solution is the Symmetric Searchable Encryption (SSE) component which will allow users' to encrypt their data using a symmetric secret key and later be able to search directly on the encrypted data. In the rest of the paper, we will be assuming the existence of the following SSE scheme as defined in [17].

Definition 1 (Dynamic Index-based SSE): A dynamic index-based symmetric searchable encryption scheme is a tuple of nine polynomial algorithms $\mathsf{SSE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{SearchToken}, \mathsf{AddToken}, \mathsf{DeleteToken}, \mathsf{Search}, \mathsf{Add}, \mathsf{Delete}, \mathsf{Dec})$ such that:

- Gen is probabilistic key-generation algorithm that takes as input a security parameter and outputs a secret key K. It is used by the client to generate her secret-key.
- Enc is a probabilistic algorithm that takes as input a secret key K and a collection of files $\mathbf{f}$ and outputs an encrypted index γ and a sequence of ciphertexts $\mathbf{c}$. It is used by the client to get ciphertexts corresponding to her files as well as an encrypted index which are then sent to the storage server.
- SearchToken is a (possibly probabilistic) algorithm that takes as input a secret key K and a keyword w and outputs a search token $\tau_s(w)$. It is used by the client in order to create a search token for some specific keyword. The token is then sent to the storage server.
- AddToken is a (possibly probabilistic) algorithm that takes as input a secret key K and a file f and outputs an add token $\tau_a(f)$ and a ciphertext c_f. It is used by the client in order to create an add token for a new file as well as the encryption of the file which are then sent to the storage server.
- DeleteToken is a (possibly probabilistic) algorithm that takes as input a secret key K and a file f and outputs a delete token $\tau_d(f)$. It is used by the client in order to create a delete token for some file which is then sent to the storage server.
- Search is a deterministic algorithm that takes as input an encrypted index γ, a sequence of ciphertexts $\mathbf{c}$ and a search token $\tau_s(w)$ and outputs a sequence of file identifiers $\mathbf{I}_w \subset \mathbf{c}$. This algorithm is used by the storage server upon receive of a search token in order to perform the search over the encrypted data and determine which ciphertexts correspond to the searched keyword and thus should be sent to the client.
- Add is a deterministic algorithm that takes as input an encrypted index γ, a sequence of ciphertexts $\mathbf{c}$, an add token $\tau_a(f)$ and a ciphertext c_f and outputs a new encrypted index γ' and a new sequence of ciphertexts $\mathbf{c}'$. This algorithm is used by the storage server upon receive of an add token in order to update the encrypted index and the ciphertext vector to include the data corresponding to the new file.
- Delete is a deterministic algorithm that takes as input an encrypted index γ, a sequence of ciphertexts $\mathbf{c}$ and a delete token $\tau_d(f)$ and outputs a new encrypted index γ' and a new sequence of ciphertexts $\mathbf{c}'$. This algorithm is used by the storage server upon receive of a delete token in order to update the encrypted index and the ciphertext vector to delete the data corresponding to the deleted file.
- Dec is a deterministic algorithm that takes as input a secret key K and a ciphertext c and outputs a file f. It is used by the client to decrypt the ciphertexts that she gets from the storage server.

Problem Statement: *Let $\mathcal{U} = \{u_1, \ldots, u_n\}$ be the set of authorized users of the CSP. We assume that a user u_i wishes to store a file f in the storage cloud provided by CSP. The problem is how to achieve the following:*

1) *u_i must be able to select a set $\mathcal{T}_i$ of locations that are considered as trusted and in which f should be stored;*
2) *u_i must have the option to validate the location of f at any time;*
3) *Even if the CSP acts maliciously and stores replicas of data in a location $l_m \notin \mathcal{T}_i$, u_i must be able to store f in such a way that CSP will not be able to learn any valuable information about the content of f;*

Adversarial Model: Similar to existing works in the area, we make the following assumptions regarding the threat model we consider. First, we assume physical security of the CSP as well as of the devices that are used by the users to send and retrieve data to the cloud. Furthermore, we assume that

the communication channel between a user' u_i and the CSP is secure. Thus, all the communication from and to the CSP is considered secure. In addition to that, we also assume that all cryptographic operations that are used throughout the protocol are semantically secure, and an adversary is not able to break any cryptographic mechanism. Finally, we assume that the adversary is acting under the *semi-honest* threat model. In the semi-honest adversarial model, even corrupted entities correctly follow the protocol specifications. However, adversaries overhear all messages and may attempt to use them in order to learn information that otherwise should remain private.

IV. LocLess

In this section, we introduce our protocol which satisfies the criteria mentioned in the problem statement and offers secure storage functionality for the users' of a cloud-based application that stores their personal records in the cloud. Before we proceed with the actual description of the protocol we provide a high-level overview of the phases that our protocol consists. Figure 1 contains a high-level representation of the main functions that our protocol consists of (details have been omitted for clarity).

The protocol considers a cloud service provider which uses a set of hosts distributed in different geographic locations. Hence, CSP can transfer data to any possible remote location without asking for the permission of the data owner. Our protocol is divided into *four main phases*: the registration and login phase, the key generation phase, the secure placement of data in the cloud by a user and finally the retrieval of data by user in a privacy-preserving manner.

Registration: Before accessing cloud data, a new user' first needs to register. To do so, the registration phase requires the user to contact the CSP and submit her identity. Then, CSP is responsible for verifying the validity of the user and can also prevent a user from creating multiple accounts/identities by simply checking if a user with the same identity has already created an account. After the verification process, user' can login to her account by simply using a standard client where will have to provide her credentials. During the first login, user' will have to select a list of possible locations that would like to store her data. To this end, CSP sends a list with all geographically distributed hosts that owns. So, CSP contacts u_i by sending $m_1 = \langle t, u_i, \mathcal{L} = \{l_1, \ldots, l_n\}, \sigma_{CSP}(u_i||\mathcal{L})\rangle$, where t is a timestamp, $\mathcal{L}$ is the set of all possible locations that CSP can store data, $\sigma_{CSP}(u_i||\mathcal{L})$ is a signature of the unique id of the user' concatenated with the list of geographically distributed hosts that are managed by the CSP. Upon reception of m_1, u_i first checks the timestamp in order to verify the freshness of the message. Then, uses the public key of the CSP to verify the signature and the the integrity of the actual message. If this verification is correct, then proceeds in selecting a list $\mathcal{S}_{T_i} \subseteq \mathcal{L}$ of locations out of those in $\mathcal{L}$ who wishes to store her files in the future. As a next step, u_i verifies the actual geographical location of the selected hosts by running a distance bounding protocol such as the one describe in [18]. After verifying the location of the selected hosts, u_i contacts CSP by sending $m_2 = \left\langle t', \mathcal{S}_{T_i} = \left\{s_1^{T_i}, \ldots, s_l^{T_i}\right\}, \sigma_i(u_i||\mathcal{S}_{T_i})\right\rangle$. Upon reception of m_2, CSP verifies the integrity and the freshness of the message and configures a set of VMs operated by hosts that are located in a subset of $\mathcal{S}_{T_i}$. From this point, u_i will be able to start storing data on remote locations that are managed by the CSP.

Store Data: After the successful selection of possible remote locations for storing data, u_i can start interacting with the CSP for storing and retrieving data from the corresponding hosts. As we described in Section I-A, this can be done with two different ways.

Scenario 1: First, we consider the case where u_i wishes to protect her data from any kind of unauthorized access. Even if the cloud service provider transfers the data or replicas of the data in a location outside of $\mathcal{S}_{T_i}$ u_i needs to be sure that no one else apart from herself will be able to read the actual content of the stored data. To achieve that, u_i will be using a SSE scheme to encrypt the data before sending it to the CSP. To do so, u_i needs to have a unique symmetric encryption key that will be used to keep her data hidden from any potential attacker. Thus, before start sending data to the CSP, u_i executes $\mathsf{K_i} \leftarrow \mathsf{Gen}(1)$ to generate a symmetric secret key. $\mathsf{K_i}$ will be used to encrypt user's private data. After the successful generation of $\mathsf{K_i}$, u_i is now ready to store encrypted data to the CSP. Lets assume that u_i wants to securely store a collection of files $\mathbf{f_i}$ to the storage offered by the CSP. To do so, u_i executes $(\gamma_i, \mathbf{c_i}) \leftarrow \mathsf{Enc}(\mathbf{f_i}, \mathsf{K_i})$ and outputs a collection of ciphertexts $\mathbf{c_i}$ as well as an encrypted index γ_i. Both γ_i and $\mathbf{c_i}$ are then sent to the CSP via a secure channel. Upon reception, CSP stores $\mathbf{c_i}$ along with the encrypted index γ_i in a local database. Since the encryption has been taken place by the user' and without the interaction of the CSP and we have assumed that user's machine is not compromised then the CSP or any other internal or external attacker will not be able to extract any valuable information about the content of the encrypted file as long as the key is $\mathsf{K_i}$ is secure.

Scenario 2: Second, we consider the simple scenario where u_i does not necessarily wish to protect her data from unauthorized access. Meaning that the actual location of the underlying server is irrelevant. This scenario is straightforward and can be done with any standard ways of uploading files to the cloud. However, in LocLess u_i will still have the ability to check the location where her files are stored. More precisely, we assume that u_i wishes to store a collection of files $\mathbf{f_j}$ to the storage offered by the CSP. To do so, u_i sends $\mathbf{f_i}$ over a secure channel to the CSP along with $\mathcal{S}_{T_i}$. Upon reception, CSP stores the received files as well as replicas in different locations from the set $\mathcal{S}_{T_i}$. Then, u_i has the option to verify the location of the files by running a distance bounding protocol such as the one described in [18]. However, in this scenario we make two major assumptions: *(1)* CSP is trusted – meaning that will not lie about the number of generated replicas and *(2)* that the distance bounding protocol cannot be hijacked. Nevertheless, both of these assumptions can be considered as unrealistic since they weaken the actual threat model that we have assumed. Having though in mind that the collection of files $\mathbf{f_j}$ stored by u_i are not considered as confidential, our protocol is sound.

Search Over the Encrypted Data: Now that u_i has stored a collection of files in the cloud storage she can start searching directly over her encrypted data. Lets assume that u_i wishes

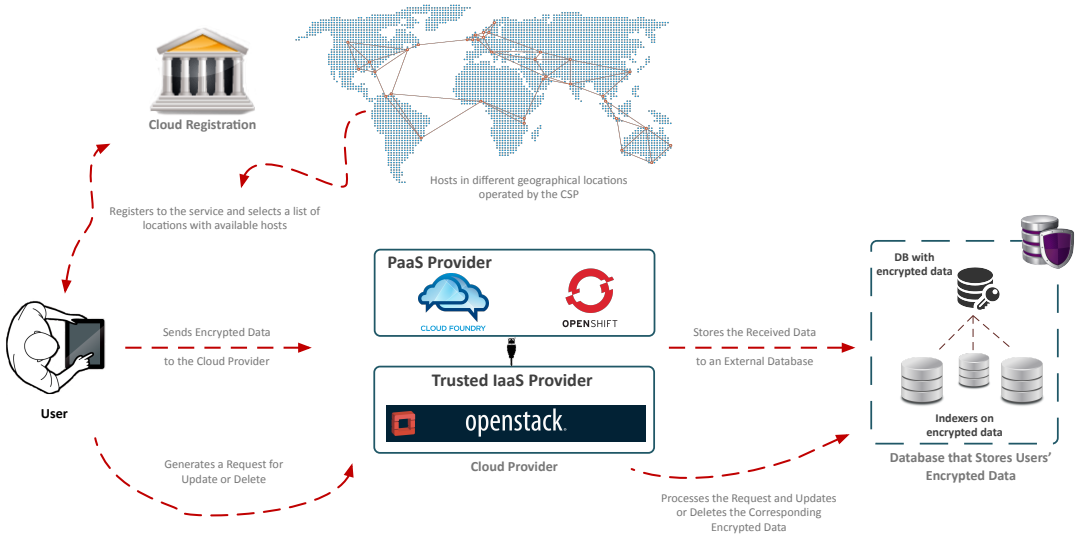

Fig. 1. High level overview LocLess

to search over her data for a specific keyword w. First, u_i executes the $\tau_s(w) \leftarrow \mathsf{SearchToken}(\mathsf{K_i}, w)$ and outputs a search token $\tau_s(w)$ that is sent to the CSP. Upon reception, CSP executes $\mathsf{Search}(\gamma_i, \mathbf{c_i}, \tau_s(w)) \rightarrow \mathbf{I}_w$ and outputs a sequence of file identifiers $\mathbf{I}_w$ which is a subset of $\mathbf{c_i}$ and contains a list of ciphertexts that includes the keyword w. The resulted $\mathbf{I}_w$ is sent back to u_i. Upon reception, u_i executes the Dec algorithm by giving as input her secret key and the sequence of encrypted files that corresponds to the list of identifiers that received from the CSP. By doing this, u_i gains access to the plaintext of data that contains the keyword w.

Update Stored (Encrypted) Data: Apart from storing data and searching over the encrypted data, the user also needs to update her stored data. Here, we consider the scenario where u_i wishes to add a new file f to the cloud storage. A naive approach that u_i could follow would be to run Enc algorithm again, generate the ciphertext of f and send it to the CSP. However, this would mean that u_i would also create a new encrypted index that would correspond to the encryption of file f. Such an approach is not efficient since the user would end-up with a huge list of encrypted indexes that are not related to each other and every time that wishes to perform a search over her data would require from the CSP to search over all the encrypted indexes. To avoid this, u_i needs to store her new file and instead of creating a separate encrypted index she needs to update the current one in order to also include the newly added file. To achieve that, u_i first generates an add token by executing $(\tau_\alpha(f), c_f) \leftarrow \mathsf{AddToken}(\mathsf{K_i}, f)$ and sends it to the CSP. Upon reception, CSP executes $\mathsf{Add}(\gamma_i, \mathbf{c_i}, \tau_\alpha(f), c_f) \rightarrow (\gamma', \mathbf{c_i'})$ and outputs an updated encrypted index γ_i' and an updated sequence of ciphertexts $\mathbf{c_i'}$ that corresponds to the data stored by u_i. Thus, by running the Add algorithm, CSP stores the ciphertext of f and updates the existing encrypted index and ciphertext list of u_i.

Delete Stored Data: The final operation that u_i needs to be able to execute, is the deletion of a file. Lets assume that u_i wishes to delete the file f that stored in the previous step. Similar to the previous case, the deletion of a file will also require the update of the existing encrypted index as well as the sequence of stored ciphertexts. To this end, u_i generates a delete token by executing $\tau_d(f) \leftarrow \mathsf{DeleteToken}(\mathsf{K_i}, f)$. Then, u_i sends $\tau_d(f)$ to the CSP who executes $\mathsf{Delete}(\gamma_i', \mathbf{c_i'}, \tau_d(f)) \rightarrow (\gamma_i'', \mathbf{c_i''})$. Similar to the Add algorithm, Delete after removing the requested file f then updates both the corresponding encrypted index and the sequence of ciphertexts that are related to user u_i.

V. Security Analysis

In this section, we are discussing the security of LocLess. The discussion focuses explicitly on the protection of data that can be accessed by unauthorized parties when they are placed maliciously, or without the actual agreement of the user', in locations that are not included in the trusted locations $\mathcal{S}_{T_i}$ that were defined by the user' during the registration phase.

The actual security of LocLess is solely based on the symmetric searchable encryption scheme that is used. More precisely, as long as the underlying cryptosystem is secure LocLess can effectively protect users' private data from unauthorized access. More precisely, the user' sends the files to the CSP in an encrypted form and the encryption has taken place by the user' without the interaction of the CSP. In addition to that, we have assumed that user's machine is not compromised. In other words, we have assumed that the symmetric key $\mathsf{K_i}$ that is used to protect user's data is secure and will never leave the perimeter of the user' or exposed to any malicious entity. As a result, even if the CSP acts maliciously and creates replicas of data that are placed in locations other than the ones in $\mathcal{S}_{T_i}$ that were defined by the user' then user's data is protected from both internal and external attacks since any attacker that can access the stored ciphertexts will not be able to decrypt the data even if she collaborates with the CSP.

Although LocLess manages to protect users' data by considering the actual location as irrelevant to the actual security of the data our approach has limitations. More precisely, the use of symmetric searchable encryption sacrifices the powerful processing capability brought by the cloud computing technology

since very limited number of queries can be implemented over encrypted data [12]. Moreover, the allowed queries are also time consuming and in many cases requires lot of computation power. Thus making LocLess difficult to use with devices that rely on limited resources (e.g. limited battery life). However, we hope that with the future developments and improvements of searchable encryption schemes LocLess will offer a significant experience to the cloud users'. Finally, this can lead to DoS [19]–[22] attacks. However such attacks are out of the scope of this work.

VI. Conclusion

In this paper, we proposed LocLess, a protocol that allows cloud users' to store their data on the cloud in such a way that location of the underlying servers will not affect their privacy. LocLess is based on a symmetric searchable encryption scheme and protects users' data from both internal and external attacks. As future steps, we plan to implement our protocol in order to measure its performance and prove its effectiveness in a real cloud environment.

As future steps, we plan to implement our protocol in order to measure its performance and prove its effectiveness in a real cloud environment. Furthermore, we plan to explore the incorporation of our protocol with mobile sensing applications and more precisely with privacy preserving reputation systems for cloud-based participatory sensing applications. The envisioned system will be based on [23]–[26] and will effectively maintain the privacy and anonymity of users' [27]–[30].

References

[1] A. Michalas, N. Paladi, and C. Gehrmann, "Security aspects of e-health systems migration to the cloud," in *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*, pp. 212–218, IEEE, 2014.

[2] A. Michalas and M. Bakopoulos, "Secgod google docs: Now i feel safer!," in *Internet Technology And Secured Transactions, 2012 International Conference for*, pp. 589–595, Dec 2012.

[3] G. J. Watson, R. Safavi-Naini, M. Alimomeni, M. E. Locasto, and S. Narayan, "Lost: Location based storage," in *Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop*, CCSW '12, (New York, NY, USA), pp. 59–70, ACM, 2012.

[4] A. Albeshri, C. Boyd, and J. G. Nieto, "Geoproof: Proofs of geographic location for cloud computing environment," in *Proceedings of the 2012 32Nd International Conference on Distributed Computing Systems Workshops*, ICDCSW '12, (Washington, DC, USA), pp. 506–514, IEEE Computer Society, 2012.

[5] A. Albeshri, C. Boyd, and J. Nieto, "Enhanced geoproof: improved geographic assurance for data in the cloud," *International Journal of Information Security*, pp. 1–8, 2013.

[6] K. Benson, R. Dowsley, and H. Shacham, "Do you know where your cloud files are?," in *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, (New York, NY, USA), pp. 73–82, ACM, 2011.

[7] M. Gondree and Z. N. Peterson, "Geolocation of data in the cloud," in *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, (New York, NY, USA), pp. 25–36, ACM, 2013.

[8] Z. N. J. Peterson, M. Gondree, and R. Beverly, "A position paper on data sovereignty: The importance of geolocating data in the cloud," in *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'11, (Berkeley, CA, USA), pp. 9–9, USENIX Association, 2011.

[9] K. Bowers, M. van Dijk, A. Juels, A. Oprea, and R. Rivest, "How to tell if your cloud files are vulnerable to drive crashes," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 501–514, ACM, 2011.

[10] R. Dowsley, A. Michalas, and M. Nagel, "A report on design and implementation of protected searchable data in iaas," tech. rep., Swedish Institute of Computer Science (SICS), 2016.

[11] A. Juels and B. S. Kaliski, Jr., "Pors: Proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, (New York, NY, USA), pp. 584–597, ACM, 2007.

[12] J. Li, A. Squicciarini, D. Lin, S. Liang, and C. Jia, "Secloc: Securing location-sensitive storage in the cloud," in *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*, SACMAT '15, (New York, NY, USA), pp. 51–61, ACM, 2015.

[13] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, (New York, NY, USA), pp. 89–98, ACM, 2006.

[14] N. Paladi, A. Michalas, and C. Gehrmann, "Domain based storage protection with secure access control for the cloud," in *Proceedings of the 2014 International Workshop on Security in Cloud Computing*, ASIACCS '14, (New York, NY, USA), ACM, 2014.

[15] N. Paladi, C. Gehrmann, and A. Michalas, "Providing user security guarantees in public infrastructure clouds," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[16] Y. Verginadis, A. Michalas, P. Gouvas, G. Schiefer, G. Hbsch, and I. Paraskakis, "Paasword: A holistic data privacy and security by design framework for cloud services," in *Proceedings of the 5th International Conference on Cloud Computing and Services Science*, pp. 206–213, 2015.

[17] A. Michalas and R. Dowsley, "Towards trusted ehealth services in the cloud," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pp. 618–623, Dec 2015.

[18] C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun, "Distance hijacking attacks on distance bounding protocols," in *2012 IEEE Symposium on Security and Privacy*, pp. 113–127, May 2012.

[19] A. Michalas, N. Komninos, N. R. Prasad, and V. A. Oleshchuk, "New client puzzle approach for dos resistance in ad hoc networks," in *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference*, pp. 568–573, IEEE, 2010.

[20] A. Michalas, N. Komninos, and N. R. Prasad, "Mitigate dos and ddos attack in mobile ad hoc networks," *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 3, no. 1, pp. 14–36, 2011.

[21] A. Michalas, N. Komninos, and N. Prasad, "Multiplayer game for ddos attacks resilience in ad hoc networks," in *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*, pp. 1–5, Feb 2011.

[22] A. Michalas, N. Komninos, and N. R. Prasad, "Cryptographic puzzles and game theory against dos and ddos attacks in networks," *International Journal of Computer Research*, vol. 19, no. 1, p. 79, 2012.

[23] T. Dimitriou and A. Michalas, "Multi-party trust computation in decentralized environments," in *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pp. 1–5, May 2012.

[24] T. Dimitriou and A. Michalas, "Multi-party trust computation in decentralized environments in the presence of malicious adversaries," *Ad Hoc Networks*, vol. 15, pp. 53 – 66, 2014.

[25] A. Michalas, T. Dimitriou, T. Giannetsos, N. Komninos, and N. Prasad, "Vulnerabilities of decentralized additive reputation systems regarding the privacy of individual votes," *Wireless Personal Communications*, vol. 66, no. 3, pp. 559–575, 2012.

[26] A. Michalas, V. A. Oleshchuk, N. Komninos, and N. R. Prasad, "Privacy-preserving scheme for mobile ad hoc networks," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pp. 752–757, June 2011.

[27] A. Michalas and T. Giannetsos, "The data of things: Strategies, patterns and practice of cloud-based participatory sensing," in *Proceedings of the 1st International Conference on Innovations in InfoBusineess and Technology*, 2016.

[28] A. Michalas and N. Komninos, "The lord of the sense: A privacy preserving reputation system for participatory sensing applications," in *Computers and Communication (ISCC), 2014 IEEE Symposium*, pp. 1–6, IEEE, 2014.

[29] A. Michalas, M. Bakopoulos, N. Komninos, and N. R. Prasad, "Secure and trusted communication in emergency situations," in *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, pp. 1–5, May 2012.

[30] K. Yigzaw, A. Michalas, and J. Bellika, "Secure and scalable statistical computation of questionnaire data in r," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2016.