Journal of Mobile Multimedia, Vol. 8, No.4 (2013) 265-286
© Rinton Press

PROVIDING A DATA LOCATION ASSURANCE SERVICE FOR CLOUD STORAGE ENVIRONMENTS

ALI NOMAN[1] CARLISLE ADAMS[2]

School of Electrical Engineering & Computer Science (EECS), University of Ottawa
[1]anoma033@uottawa.ca , [2]cadams@eecs.uottawa.ca

Received December 1, 2012
Revised May 15, 2013

In the cloud storage environment, the geographic location of the data has profound impacts on its privacy and security; it is due to the fact that the data stored on the cloud will be subject to the laws and regulations of the country where it is physically stored. This is one of the main reasons why companies that deal with sensitive data (e.g., health related data) cannot adopt cloud storage solutions. In order to ensure the rapid growth of cloud computing, we need a data location assurance solution which not only works for existing cloud storage environments but also influences those companies to adopt cloud storage solutions. In this paper, we present a Data Location Assurance Service (DLAS) solution for the well-known, honest-but-curious server model of the cloud storage environment; the proposed DLAS solution facilitates cloud users not only to give preferences regarding their data location but also to receive verifiable assurance about their data location from the Cloud Storage Provider (CSP). This paper also includes a detailed security and performance analysis of the proposed DLAS solution. Unlike other solutions, the DLAS solution allows a user to give a negative location preference regarding his/her data and works for CSPs (e.g., Windows Azure) that practice geo-replication of data (to ensure availability of data in case of natural disasters). Our proposed DLAS solution is based on cryptographic primitives such as zero knowledge sets protocol and ciphertext-policy attribute based encryption. According to the best of our knowledge, we are the first to propose a non-geolocation based solution of this kind.

Key words: Cloud computing security, proof of data location, proof of storage, data location assurance service for cloud, accountable cloud
Communicated by: A. Hakim, M. Agoulmine, & G. Parr

1 Introduction

In 1961, John McCarthy, the computer scientist and inventor of the term “Artificial Intelligence”, made a prediction that “Computing may someday be organized as a public utility, just as the telephone system is a public utility [1].” The introduction of cloud computing takes us very close to that prediction. The basic idea behind cloud computing is that a cloud user (an organization or an individual) can use computation and storage resources on demand from a virtualized environment known as a “cloud” using the “pay as you go” approach. Rather than owning (and maintaining) a large and expensive IT infrastructure, a customer now has the alternative to rent the necessary resources from the cloud whenever he/she needs them; that is why the popularity of cloud computing is increasing day by day [2].According to a very recent report (which forecasted the cloud computing market for USA) published in ITCandor, In USA cloud computing will continue to grow. Whereas

$293 billion is expected to be spent in this market by 2013, by 2017 the total spending on cloud computing market will be reached to $421 billion [3].

However security has been identified as arguably the most significant barrier for the faster and more widespread adoption of cloud computing [4]. In a cloud computing environment, a user has very little or no control over their resources kept in the cloud; as a result, he or she has to rely on the cloud service provider for the "well being" of his/her resources. This unique characteristic of cloud environments make it very challenging for researchers to come up with security and privacy solutions for cloud environments.

"*In a cloud environment, a user does not know where his data is physically located* "- this is one of the security issues in cloud environments which need addressing since it creates a number of major security concerns for cloud users [5]. Like other works [5, 6], we agree with this concern and in our previous short paper [7], we name this security issue as the "*data location problem*" of cloud storage environments. This paper focuses on finding a solution for the data location problem of cloud storage environments.

The geographic location of the data has significant effects on its confidentiality and privacy; the reason is that the data stored on the cloud will be subject to the laws and regulations of the country where it is physically stored [8]. More precisely, health and personal privacy laws of a country might enforce restrictions on the physical location of the data. For example, the Canadian Personal Information Protection and Electronic Documents Act [9] requires data to be stored within Canada and so any outsourcing of data storage must be with a guarantee about the location of the stored data.

In order to ensure the rapid growth of the cloud storage business, we need a solution for the "data location problem" of cloud storage environments i.e., we need a data location assurance solution which should meet the requirements not only for the users (e.g., companies that deals with sensitive data) but also for the existing CSPs such as Amazon and Windows Azure. Figure 1 includes some scenarios that point out why a data location assurance solution is required from a user's perspective. Furthermore, a data location assurance solution will not be a practical one if it does not work with the CSPs (e.g., Windows Azure) that do geo-replication of their users' data in order to ensure availability of data in case of natural disasters. Considering the geo-replication practice and scenarios illustrated in Figure 1, we think a good data location assurance solution should also allow a user to give negative location preference for his/her data i.e., 'Do not keep my data in region R."

It is true that some cloud service providers (such as Amazon) allow their users to specify a preference for the geographic location of data in the Service Level Agreements (SLA). However in order to reduce IT costs, a CSP might violate this SLA intentionally or accidentally by moving users' data into a datacenter which is located outside of the specified geographic boundaries[8]. Most importantly, at present the existing CSPs do not offer any solution which could enforce the location restrictions of data. So in this case, you as a user have no means to *continuously verify* that cloud storage providers are meeting their contractual geographic obligations; by the time you become aware of a violation of this contract, your data has already been affected [10].

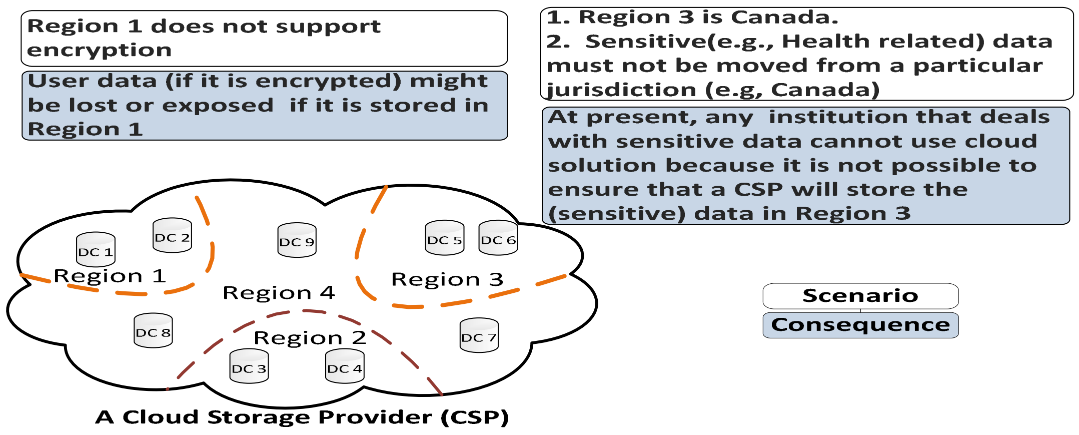
Region 1 does not support encryption
User data (if it is encrypted) might be lost or exposed if it is stored in Region 1
1. Region 3 is Canada.
2. Sensitive(e.g., Health related) data must not be moved from a particular jurisdiction (e.g, Canada)
At present, any institution that deals with sensitive data cannot use cloud solution because it is not possible to ensure that a CSP will store the (sensitive) data in Region 3
DC 1
DC 2
DC 9
DC 5
DC 6
Region 1
Region 4
Region 3
DC 8
Region 2
DC 7
DC 3
DC 4
A Cloud Storage Provider (CSP)
Scenario
Consequence

Figure 1. Possible scenarios of not having a data location assurance solution

In this paper, we propose a complete data location assurance solution for cloud storage environments which can also be considered as a service provided by a CSP; we name it DLAS: Data Location Assurance Service. The proposed DLAS solution is able to satisfy the requirements for both the existing CSPs and cloud users. It is based on two cryptographic primitives: zero knowledge sets (ZKS) protocol [11] and ciphertext-policy attribute based encryption (CP-ABE) scheme [12]. In accordance with other security solutions [13, 14, 15, 16, 17] for cloud computing, we also consider the well-known "honest-but-curious" server model for DLAS. The proposed DLAS solution allows a user to give his/her data location preference in the following form: "*Keep (or do not keep) this data in region*", where *region* can be a country or a continent. Furthermore in the DLAS solution, a user can make a data location verification request at any time and in response to his request he always receives a verifiable answer (about his/her data location) from the cloud.

1.1 Related Work:

The data location problem has been identified as one of the security issues in cloud computing by many researchers [5, 6, 10, 18]. Research conducted by Gartner also suggests that cloud users' data location preference should be included in the SLA [6]. They also recommend that users should continuously check whether their data is still located in the appropriate jurisdiction.

Geo-location techniques (e.g., ping based solution) could be used to get a 'rough' estimate of the datacenters'/server's location and thus provide *some knowledge* to the users about their data location [19]. There are some existing solutions [8, 9, 10] which incorporate geo-location based techniques to address the '*data location problem*' of cloud computing. However this geo-location technique based solutions is unable to fulfill one of the main requirements of the existing cloud storage environments which is geo-replication of data. More precisely, geo-location based solutions do not work if a piece of data is stored/copied in multiple datacenters; what if a copy of your data is stored outside your preferred location but you are getting the response of your query from a legitimate datacenter? Considering the geo-replication practice of existing cloud storage providers (e.g., Windows Azure), we are interested to come up with a DLAS solution which do not rely on geo-location technique.

To our knowledge, [10] is the first work which is related to the data location assurance solution mentioned above. In [10], Peterson et al. address the data location problem within their notion of "data sovereignty". In their position paper, they provide a direction (not a solution) to develop a 'future tool' (solution) by which a user would be able to *predict* the location of their data. Their suggested solution is based on the combination of 2 techniques: MAC-based PDP (Provable Data Possession) scheme and Geo-Location Based solution.

In our previous short paper [7], we give a rough sketch of the first (non-geolocation based) DLAS solution. However almost at the same time of our previous work [7], both Watson et al. [9] and Albeshri et al. [8] independently come up with a solution for the data location problem. Both of these solutions are based on geolocation techniques and do not fulfil the geo-replication requirement of the existing CSPs (e.g., Windows Azure). More precisely, unlike our proposed solution, these two solutions do not work if a piece of data is stored in multiple datacenters. The solution proposed in [9] is based on POR (Proof of Retrievability) and geolocaion techniques. However their underlying system model is different and the authors of that paper [9] also agree with the fact that their solution is not suitable for existing cloud storage providers such as Amazon. On the other hand, the solution proposed by Albeshri, et al. [8], is based on very restrictive assumptions about the system model; it assumes that all datacenters should have a tamper-proof GPS device and there also exists a third party auditor who continuously communicates with this device. In addition, in both of these solutions, it is the user's task to do most of the required computations; the server's task is much less. However we believe, from a user's perspective, one of the main reasons to go for a cloud solution is to transfer most of the (computation) burden, if not all, to the cloud. Furthermore these solutions are based on geolocation techniques and the analysis of existing geo-location solutions done in [19] suggests that the geo-location solution is not very accurate.

As a continuation of their previous work [10], Peterson et al. recently (in 2013) propose a 'real' solution which is based on constraint based geolocation technique and PDP [27]. Like [8, 9], this solution is also unable to fulfil the geo-replication requirement of the existing CSPs (e.g., Windows Azure).

Our proposed solution, DLAS, differs from [8, 9, 27] in many ways. Unlike those solutions, our solution is not based on any geo-location techniques and most importantly it will work even if a piece of data is stored in multiple datacenters. In contrast to [8, 9, 27], in our solution we consider the fact that in any cloud computing based solution users should have minimal computational load.

Additionally, our proposed solution, DLAS, would enhance the trust level of users towards the CSP by offering the data location assurance to the user. Similar to DLAS, there is other work [20, 21] which can also provide assurance to the cloud users in terms of data integrity (the data is intact and still available in the server) [20] and computational integrity (the computation done by the cloud server is accurate) [21]. Like [20, 21], we believe, DLAS has also the potential to be used as one of the building blocks of the "Accountable Cloud."[a]

1.2 Contributions:

The contributions of this paper are summarized as follows:

[a] Accountable cloud , a concept proposed by Haeberlen[2], means that a cloud should be made accountable to both the user and the provider

- In this paper, we propose the first non-geolocation based data location assurance service solution, DLAS, for the honest-but-curious server model where a user receives verifiable assurance from the cloud service providers at any time.
- DLAS is suitable for real cloud storage environments; it works even if a piece of data is stored in multiple datacenters. In other words, unlike existing solutions [8, 9, 27], It meets the geo-replication requirement of existing CSPs such as Windows Azure.
- DLAS also allows users to give negative location preference about their data: "Do **NOT** keep data in region R."
- As opposed to [8, 9, 27], DLAS does not require the user to perform any computation.
- DLAS can enhance the trust level of users towards the CSP by offering the data location assurance to the user; we believe DLAS has also the potential to be used as one of the building blocks of the Accountable Cloud.

Organization of the paper: The remainder of the paper is organized as follows. The system and attack model are provided in section 2. The cryptographic primitives used in this solution are introduced in section 3; we discuss our proposed solution, DLAS, in detail in section 4. In section 5, we perform an extensive analysis of our proposed solution which includes both security analysis and performance analysis of DLAS. We draw our conclusions in section 6 and provide an indication of our future work in section 7.

2 System and Attack Model

2.1 System Model

We consider a system model of a cloud computing environment which consists of 3 entities: Enterprise (E), Users (U) and Cloud Storage Provider (CSP*)* [Figure 2]. In this model, in order to give data storage facility to its users, an Enterprise establishes an SLA (Service Level Agreement) with the CSP that supports the data location assurance service. This system model is very common in real world: any (cloud) service provider (not having the storage facility) could play the role of E by establishing a SLA with any cloud storage provider(CSP) in order to offer data storage service to any individual user/ company (U). In this case, it is obvious that different U (i.e.; different companies) will have different data location preferences. Another real world example would be a university (i.e., E) interested to take storage service from a CSP for its employees and students (i.e., U). In this paper, we consider *public cloud* as the deployment model (for generality) and *IaaS* (Infrastructure as a Service) as the service model. The formal definition of public cloud and IaaS service model can be found in [22].

For our proposed DLAS solution, we consider a system model which is very close to the system model of existing cloud storage providers such as Amazon and Windows Azure Storage services. More precisely, *we assume that a CSP has an internal storage mechanism to store data in its storage servers in different datacenters and our proposed solution is independent to this internal storage mechanism*. Now we point out other key characteristics of the system model considered for the proposed DLAS solution:

- A CSP has a number of datacenters, each having a number of storage servers and those datacenters are scattered all over the world. However we restrict our discussion of proposed solution in terms of datacenters.

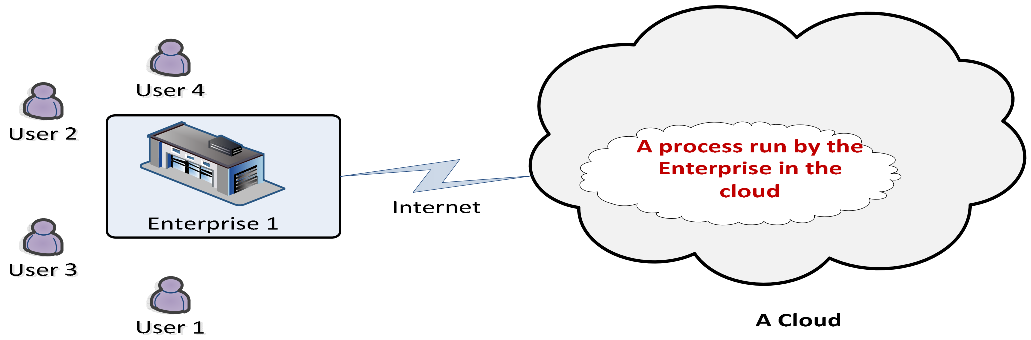

Figure 2. System Model considered for our proposed solution

- Like other real cloud storage service providers (e.g., Windows Azure [23]), our system model also considers the fact of geo-replication of data (i.e., a piece of data will be stored in multiple datacenters) in order to reduce the risk of damage due to natural disasters.
- An Enterprise can execute an arbitrary number of processes (which also includes executing encryption algorithms) in the cloud to manage its users' data. In this case, an Enterprise has total control over those processes; the CSP cannot retrieve any (meaningful) information about those processes even though they are being executed in the cloud. Note that this is a valid assumption for the IaaS service model.
- A user does not need to perform any computation; all computations are performed in the cloud.
- A user does not need to be available all the time. A user will come online only when he/she needs to either make a data storage request or verify the location of a particular data item. In all cases, a user has no direct interaction with the CSP; a user will directly interact with the processes executed by the Enterprise in the cloud. However those processes have regular interactions with the CSP.
- We assume that the CSP as well as the processes executed by the Enterprise[b] in the cloud are always available/online and they have sufficient computational power and storage capacity.
- The CSP is responsible for sending regular updates (once in a day; e.g., at 12:00 am) to the Enterprise which contains data location information for all data of an Enterprise. Many CSPs will do this because the proposed DLAS solution will enhance the trust level of users towards the CSP and eventually the CSP can attract more users; furthermore, the market of cloud computing is becoming competitive.
- All communication channels (between user and the Enterprise, and between the CSP and the Enterprise), are assumed to be secure.

2.2 Attack Model

Like other security solutions [13, 14, 15, 16, 17], we consider the well-known "honest-but-curious" Cloud Server Model. That is to say, cloud storage servers will *perform* their regular tasks correctly; however they may try to find out as much secret information as possible regarding their clients.

We consider the following attack scenarios where an adversary may act:

[b] Throughout the discussion of our proposed solution, when we say "the Enterprise" it actually means the processes run by the Enterprise in the cloud.

As the CSP: A CSP may try to store a piece of data in a location that does not satisfy user's data location preference and it tries to keep this fact undetected.

As the Enterprise: An Enterprise may try to find more information regarding the datacenters of a CSP.

3 Cryptographic Primitives Used

3.1 Zero Knowledge Sets

Zero Knowledge Sets (ZKS) is a cryptographic primitive which facilitates a polynomial time Prover to commit to an arbitrary finite set S of strings so that, later on, for any string x, he can prove whether $x \in S$ or $x \notin S$, without providing any further information about S; not even the size of S [11]. Since zero knowledge set is a special case of an elementary database (EDB), it has been explained in terms of an elementary database.

DEFINITION 1. Elementary Database (EDB): An EDB, DB is a subset of $\{0, 1\}^* \times \{0, 1\}^*$ such that if $(x, v) \in DB$ and $(x, v') \in DB'$ then v = v'. By *[DB]* we denote the support of *DB*, i.e., the set of $x \in \{0,1\}^*$ for which $\exists$v such that $(x, v) \in DB$. We denote such unique v as *DB(x)*; if $x \notin$ *[DB]* then we also write $DB(x) = \perp$ [11].

A ZKS protocol is a non-interactive protocol which relies on a public random string σ, the reference string. This reference string is polynomially long in *K*, the security parameter controlling the probability of error or successful cheating.

DEFINITION 2. A ZKS protocol has three algorithms: *Commit_ZKS (DB, σ, K)*, P*roof_ZKS (DB, K, σ, PK, SK)* and *Verify_ZKS* $(x, \pi_x, K, \sigma, PK)$ and two parties- a Prover (also known as the committer) and a Verifier. Whereas the Prover is responsible to execute the Commit_ZKS and Proof_ZKS algorithm, the job of Verifier is to perform the Verify_ZKS algorithm. Here PK, SK and π_x represent public key (i.e., the commitment of DB) and private key, and DB's proof about the value, x, respectively [11].

In our proposed solution, DLAS, the CSP and the Enterprise (E) play the role of the Prover and the Verifier respectively.

A ZKS protocol must satisfy completeness, soundness and zero-knowledge. Informally, *completeness* means that if $DB(x) = y$, the Prover should always be able to convince the Verifier of this fact. *Soundness* means that no efficient Prover should be able to produce a commitment to *DB*, a value x and two proofs π_x, $\pi^1{}_x$ that convince the Verifier of two distinct values for *DB(x)*. Finally *zero-knowledge* means that an efficient verifier learns only the values *DB(x)* from his interaction with the Prover, and nothing else (not even the size of *DB*). In particular, the Verifier does not learn the values *DB(x')* for an *x*' not queried to the Prover.

Note that, many cryptographic primitives such as Pedersen commitment scheme and collision resistant hash functions have been used in the zero knowledge set protocol. [11] provides a full description of the original ZKS protocol.

3.2 CP-ABE

A Ciphertext Policy Attribute Based Encryption (CP-ABE) scheme [12] is one kind of Attribute Based Encryption where a user's private key is associated with an arbitrary number of attributes expressed as strings; when an encryptor encrypts a message, he specifies an associated access structure (also known as access tree) over attributes. A user will only be able to decrypt a ciphertext if that user's attributes pass through the ciphertext's access structure. At a mathematical level, access structures used in this scheme are described by a monotonic "access tree", where nodes of the access structure are either threshold gates or leaves which represent attributes. Note that *AND* gates can be constructed as *n-of-n* threshold gates and *OR* gates as *1-of-n* threshold gates. Figure 3 shows an example of a simple access tree.

DEFINITION 3 : A CP-ABE scheme consists of four fundamental algorithms: Setup_CPABE(Δ), Encrypt_CPABE (*PK, M, A)* , KeyGen_CPABE (MK, S) and Decrypt_CPABE (PK,CT, Credential) where Δ, PK,M, A, MK, S, CT and Credential represent implicit security parameter, public key, plain text, master key, a set of attributes, the corresponding ciphertext of M, and private key.

Note that the CP-ABE encryption scheme is based on the bilinear map. The complete description of the CP-ABE scheme can be found in [12]. At present an implementation of the CP-ABE scheme is also publicly available in [24].

4 The Proposed Solution: DLAS

The proposed solution, DLAS, has 4 important phases: Bootstrapping Phase, Offline Phase, User Data Registration Phase, and Data Location Verification Phase. Among the four phases, the Bootstrapping phase has to be executed first; the order is not important for other 3 phases; they may coincide with each other. Whereas the User Data Registration Phase and Data Location verification Phase are online (i.e. always available for execution), the Offline phase is executed once in a day.

The *Bootstrapping phase* has to be executed prior to the start of the data storage service. In this phase, the CSP generates a fixed number of pseudonyms (i.e., datacenter ID) for each datacenter it has, aiming to group each data location preference list with similar size (i.e., having a fixed number of datacenters). A data location preference list represents a particular region (could be a country or a continent). The CSP then commits its datacenter's location information for each data location preference list it has, to the Enterprise (E) using the ZKS commitment scheme. Because of the use of ZKS commitment scheme, the *only* thing an Enterprise can do is to verify whether a particular datacenter is within a particular region or not.

The *User Data Registration* phase not only facilitates a user to give preferences regarding their data location during making a data/file (we use the two terms data and file interchangeably) storage/registration request to the cloud through the Enterprise, but also provides the necessary credential to the user which will later on enable the user to verify the location of a piece of data. A user can express his / her data location preference in either one of two ways: "***Keep my data/file in region, R***" or "***Do not keep my data/file in region, R"*** where *R* can represent a country (e.g., USA) or a continent (e.g., Europe). Depending on the CSP's internal storage policy, the CSP may store user's data/file in more than one datacenter; however those datacenters should satisfy user's choice about the data location (i.e., *R*).

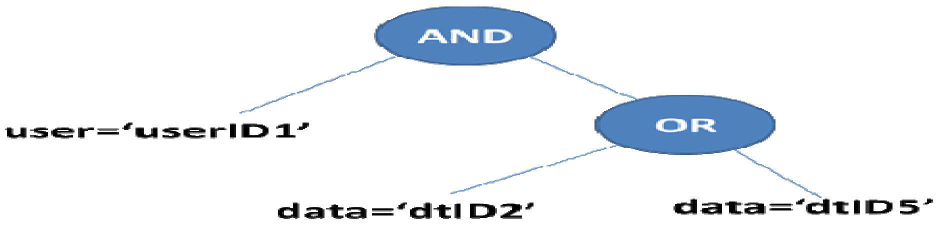

Figure 3. A simple example of access tree which allows the access if the user is *userID1* and the data is either *dtID2* or *dtID5*.

In the *Offline* phase, the Enterprise, E, builds an encrypted database (using the CP-ABE encryption scheme) from the data location information it regularly received from the CSP about all users' data of E; this encrypted database is made in such a way that later *only* a legitimate user (having the right credential) is able to decrypt and thus retrieve data location Information (i.e., datacenter id/s) only for his/her data, and thus it works as a filter by separating a valid verification request from an invalid one in the verification phase. We call this phase an offline phase because user (U) does not participate in this phase; only the CSP and the Enterprise are involved.

In the *verification phase*, users can make a verification request regarding his/her data *at any time* and get a verifiable assurance about their data location from the CSP.

Notation: The important notations of our proposed solution are listed here.

Notation Name	***Description***
U,E,CSP	*General representation of User, Enterprise and Cloud Storage Provider respectively*
userID,dataID,dcID	*Represents a User, a piece of data (e.g., a file) and a datacenter respectively*
R	*Represents user's data location preference (e.g., Europe)*
Δ,Δ'	*Implicit security parameter of the ZKS and CP-ABE protocol respectively*
σ, K,SK	*σ and K are random reference string and security parameter of ZKS protocol respectively. SK is the private key of the ZKS protocol*
PK, MK	*Public parameters and Master key of the CP-ABE protocol*
A	*Represents an access structure (can be based on attributes such as R, userID, dataID). It is an important parameter of the CP-ABE scheme.*
S	*A set of attributes; it could be dataID, dcID…*
Credential	*The output of the key generation algorithm of the CP-ABE protocol*
ρ	*Represents user inputs regarding the storage request (consists of file, selection of regions etc)*
Ω	*It may include time stamps and other parameters related to the storage policy.*
ID,	*A set of pseudo IDs generated from a real ID, id*
DCID,DCID_res	*Both represents a set of dcIDs*
DB	*An elementary database generated from a given set*
DB_commit	*An encrypted database; generated by applying the commitment algorithm of ZKS protocol on the database, DB*
userDataStorageReqDB / DB2	*A database having following attributes:*
	userID,dataID,dcID, R

data LocationDB_E/DB1	*A database having following attributes: dataID, dcID*
tempDB	*A database having following attributes: dataID,userID, dcID,R*
dataLocationFinderDB	*A database generated by encrypting each row of tempDB*

4.1 DLAS Scheme-Formal definition

Here we formalize the notion of DLAS, with the system and attack model outlined in section 2.

DEFINITION 4. A DLAS scheme consists of 4 phases.

Phase 1: Bootstrapping Phase: This phase consists of 5 algorithms.

Setup_Securityparameter_ZKS (Δ) → (σ, K)*: is run by the CSP. It takes Δ (implicit security parameter) as input and outputs σ (a random reference string) and K (the security parameter). The output of this algorithm is publicly available.*

Setup_CPABE (Δ′) → (PK, MK): *is run by the Enterprise, E. It takes Δ′ (implicit security parameter) as input and generates the public parameters PK and master key MK.*

Generate_pseudoID (id, n) → ID (a set of pseudo IDs): *is run by the CSP. It takes the real id and n (number of pseudo IDs that need to be generated) as a parameter and outputs ID, a set having n pseudo IDs.*

Convert_set_to_elementaryDatabase (S) → (DB): *is run by the CSP. It takes a set (e.g., S = {x1, x2, x3, …}) as input and generates an elementary database, DB as output (where each row of DB is in the form of <xi, 1>). For each data location preference list, the CSP executes this algorithm.*

Commit_ZKS (DB, σ, K) → (DB_commit, SK): *is run by the CSP. It is the commitment algorithm of the ZKS protocol. This algorithm takes elementary database, DB, σ, K as inputs and generates the commitment, DB_commit (which could be also considered as the public key) for DB and the private key, SK. The CSP generates this commitment (i.e., DB_commit) for each data location preference database and sends them to the Enterprise, E.*

Phase 2: User Data Registration Phase: This phase consists of 3 algorithms. This phase starts with a data/file storage request having a location preference, R, made by a user (U) to the CSP through the Enterprise, E. This results in the file to be stored in one or more of the datacenters of the cloud storage provider.

GenStorageReqParam (ρ) → (dataID, R, Ω): *The Enterprise runs this algorithm once it receives the storage request from its user. This algorithm takes ρ (user inputs regarding the storage: file, selection of regions) as inputs and generates all necessary parameters (such as dataID, a set of data location preferences, R and Ω (which may include time stamps and other parameters related to the storage policy).After storing this storage request information in the database DB2, E forwards the storage request (E→CSP: <dataID, file, R, Ω >) to the CSP.*

FindDataCenters(R, Ω) → DCID: *is run by the CSP. It takes R (a set of data location preferences) and Ω (internal storage policy) as inputs and outputs DCID, a set of dcIDs (datacenter id), where the user's file will be stored.*

KeyGen_CPABE (MK, S) → (Credential): *It is the key generation algorithm of the CP-ABE scheme which is run by E. This algorithm takes the master key MK and a set of attributes, S (which may include userID, dataID, R, timestamp) as inputs and outputs the Credential,, which will be sent to the user.*

Phase 3: Offline Phase: This stage consists of 2 algorithms. This phase starts with the CSP who regularly sends the updated database, DB1 (which includes <dataID, dcID> information of the Enterprise, E) to the E. The Enterprise then builds an encrypted database using DBI and other information about its users.

CheckUpdate (DB1, prev_Hash) → $a \in \{yes, no\}$: *is run by the CSP. It takes DB1 and prev_Hash (i.e., previous hash of the DB1) as inputs and determines whether there is an update/change in the database which needs to be send to the E.*

BuildEncryptedDB (DB1, DB2) → DB3: *is run by the E. It takes database DB1 and DB2 (which includes <userID, dataID, R> as inputs and generates an encrypted Database, DB3, using the encryption algorithm of the CP-ABE scheme.*

Pseudocode:

```
1. tempDB←select userID, DB1.dataID, dcID, R from DB1, DB2 where
   DB1.dataID=DB2.dataID
2. For each row(ri) in tempDB
   2.1.    A←set the access structure based on  attributes such as
           R, userID, dataID of ri
   2.2.    ri'(i-th row of DB3) ←Encrypt_CPABE(PK, ri, A)
```

Phase 4: Data Location Verification Phase: This stage consists of 2 algorithms. This phase starts with a data location verification request made by the user and ends with producing a verifiable assurance regarding the user's data location (if the request is valid).

VerifyRequest (DB3, userID, dataID, R, credential) → *DCID_res or ⊥: is run by the Enterprise, E. It takes userID, dataID, R and* *credential (SK) as inputs and outputs DCID_res (a set of dcIDs) if the data location verification query is valid; otherwise it returns* $\perp$.

```
Pseudocode:
1. DCID_res←{}
2. count←0
3. validUser←false
4. For each row(ri) in DB3
   4.1.     temp←Decrypt_CPABE(PK, ri, credential)
   4.2.     if (temp) //decryption is successful
      4.2.1.      validUser←true
      4.2.2.      dcIDtemp← select dcID from ri
      4.2.3.      DCID_res← DCID_res ∪ {dcIDtemp}
      4.2.4.      count←count +1
5. if (NOT validUser)

   5.1.     return ⊥

6. if (count>0)
      Return DCID_res
   else if (count = 0)
      "This data item is no longer available in the CSP"
```

Note that the decryption process of any row *i of* the database, *DB3* will be successful if the user's credential satisfies the access structure of row *i*. In other words, a user will get *dcID* (or *dcIDs* if the data is stored in multiple datacenters) only for his/her data.

VerifyDataLocation (DCID_res, DB_commit_DBs, R) *←a* $\in$*{yes, no}: is run jointly by the CSP and the Enterprise. It takes DCID_res, user's preferred storage location/s R and DB_commit_DBs (commitments for all location preference list DB) as inputs and produces a verifiable answer (in Boolean form) regarding user's data location in a given region.*

```
Pseudocode:
1. For each dcID in DCID_res
   1.1.     For each database,DB_commit in DB_commit_DBs
      1.1.1.      CSP generates a proof, $\pi_x$ for dcID (dcID ∈
            DB_commit or dcID ∉ DB_commit) by executing Proof_ZKS
            (DB, K, $\sigma$, PK, SK) and sends it to E.
      1.1.2.       E verifies the membership/non-membership proof, $\pi_x$
            by executing Verify_ZKS( x, $\pi_x$ , K, $\sigma$ , PK):
            result ←Verify_ZKS(x, $\pi_x$ , K, $\sigma$ , PK)

      1.1.3.      If ((result AND dcID ∈ DB_commit AND DB_commit Not
            IN R) OR (result AND dcID ∉ DB_commit AND DB_commit  IN
            R))
         1.1.3.1.      Return false
      1.1.4.   Return true
```

4.2 DLAS Construction

Here we describe how a DLAS scheme can be constructed. This DLAS construction heavily relies on two cryptographic primitives: the ZKs protocol and the CP-ABE encryption scheme.

4.2.1 Bootstrapping Phase:

In this phase, the CSP commits its datacenters' location information for each data location preference list it has, to the Enterprise (E); here a data location preference list represents a particular region (could be a country or a continent). However there are some other tasks involved in this phase. Figure 4 provides an overview of the Bootstrapping phase. Whereas both the CSP and the Enterprise participate in this phase; most of the tasks are performed by the CSP. This phase works as follows:

Step 1: CSP executes the `Setup_Securityparameter_ZKS()` algorithm to generate necessary security parameters for the ZKS scheme. The Enterprise, E, also generates public parameters, PK, and master key, MK, by executing the setup algorithm of the CP-ABE scheme which is `Setup_CPABE()`.

Step 2: CSP creates a fixed number of pseudo IDs for each (actual) datacenter (dcID) by executing the *`Generate_pseudoID()`*. The reason for doing this is to minimize the reuse of the same dcID so that it would be more difficult for both the user and the Enterprise to make any prediction regarding the datacenter's location without participating in the ZKS protocol.

Step 3: For each data location preference list, CSP creates an elementary database (EDB), DB, using the `Convert_set_to_elementaryDatabase()` algorithm. A simple way of creating an EDB from a given set is to use each element (in this case it is dcID) of the set as a key and assign an arbitrary value (e.g., 1) for each key. If we assume that a CSP has N data location preference lists for N Regions, there will be N EDBs.

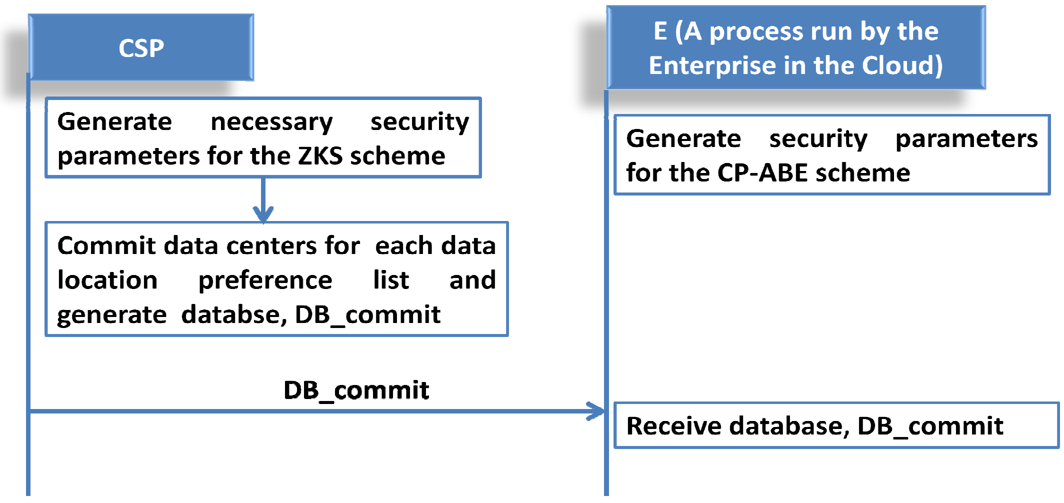
CSP
E (A process run by the Enterprise in the Cloud)
Generate necessary security parameters for the ZKS scheme
Generate security parameters for the CP-ABE scheme
Commit data centers for each data location preference list and generate databse, DB_commit
DB_commit
Receive database, DB_commit

Figure 4. The Bootstrapping Phase

Step 4: At this point CSP executes the `Commit_ZKS()` algorithm to generate a commitment for each database, DB, generated in step 3; we call this database, DB_commit. This commitment algorithm also produces associated private key SK, which will later be used by the CSP in the data location verification phase.

Step 5: The CSP sends all *DB_commit* databases to E. Note that whenever there is a change (i.e., a new datacenter is added into or removed from a particular region, R) in any data location preference list, the CSP must commit the updated preference list to the Enterprise.

4.2.2 User Data Registration Phase

User, Enterprise and CSP participate in this phase. In our model User cannot directly contact CSP; Enterprise resides in the middle between User and CSP. Figure 5. provides an overview of this phase. This phase works as follows:

Step 1: This phase starts when a user makes a data/file storage request with a particular storage location preference, R. After receiving the storage request from the user, the Enterprise executes the `GenStorageReqParam()` algorithm in order to generate necessary parameters (it might include dataID, user's location preference R, timestamp and so on) required for sending the real storage request to the CSP.

Step 2: The Enterprise, E, first stores all necessary information regarding the user's storage request in the database, *userDataStorageReqDB.* [Note that the CSP has no control over the databases created by E since we consider IaaS as the deployment model.] It then forwards the storage request to the CSP

E→CSP: <dataID, R, Ω *>*

Step 3: Upon receiving the storage request from step 2, the CSP executes the `FindDataCenters()` algorithm to find a set of datacenters (i.e. dcIDs) where the file can be stored. Upon successful storage, the CSP firsts updates its database dataLocationDB_E (<dataID, dcID>) and then sends an OK message to the Enterprise.

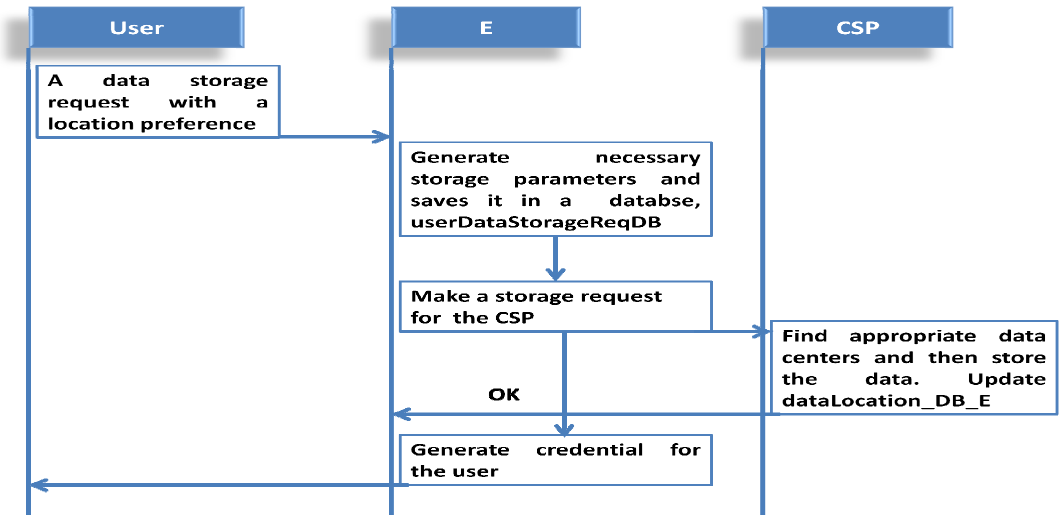
User
E
CSP
A data storage request with a location preference
Generate necessary storage parameters and saves it in a databse, userDataStorageReqDB
Make a storage request for the CSP
Find appropriate data centers and then store the data. Update dataLocation_DB_E
OK
Generate credential for the user

Figure 5. The Data Registration Phase

We assume that the CSP ensures the correctness of the database, dataLocationDB_E (i.e., are these data items really stored in those specified datacenters?) prior to sending it to the Enterprise.

Step 4: After receiving the OK message from the CSP, the Enterprise generates a credential by executing the `GenCredentials()` algorithm. It then sends the credential to the user as a response to this/her storage request.

E → U: <userID, dataID, credential, R>

This credential later on enables users to verify their data location.

4.2.3 Offline Phase

The CSP and the Enterprise, E take part in this phase. Figure 6. shows an overview of this phase. The Offline phase works as follows:

Step 1. The CSP first executes the `CheckUpdate ()` algorithm in order to find out whether there is any change in the database *datalocationDB_E.* If the algorithm returns *yes*, the CSP sends the updated *datalocationDB_E* to E.

Step2. After receiving the database *datalocationDB_E*, the Enterprise executes the (`BuildEncryptedDB (`*`datalocationDB_E, userDataStorageReqDB`*`)→` *`dataLocationFinderDB`*) algorithm to build an encrypted database, *dataLocationFinderDB*

4.2.4 Data Location Verification Phase

User, Enterprise and the CSP all participate in this phase. Figure 7. gives an overview of this phase. This phase works as follows:

Step 1: This phase starts when a user makes a data location verification request to the CSP through the Enterprise. Since the Enterprise, E, works as a proxy/middleman between User and CSP, the Enterprise receives the data location verification request.

U →E: (userID, dataID, credential, R)

Step 2: At this point, the Enterprise, E executes the [`VerifyRequest(`*`dataLocationFinderDB`*`, userID, dataID, R, credential)`] algorithm and finds out (i) whether the verification request is valid and (ii) if so, finds DCID_res, a set of all datacenter Ids where that particular data is stored.

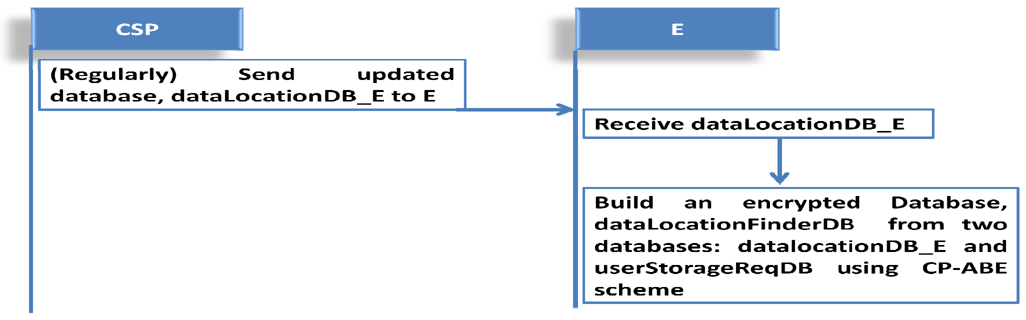
CSP
E
(Regularly) Send updated database, dataLocationDB_E to E
Receive dataLocationDB_E
Build an encrypted Database, dataLocationFinderDB from two databases: datalocationDB_E and userStorageReqDB using CP-ABE scheme

Figure 6. The Offline Phase

Note that this algorithm will also notify the Enterprise and a user, if a particular data item is no longer available in the CSP (e.g., this data item was accidentally deleted from the storage server/s).

Step 3: The Enterprise then executes the [`VerifyDataLocation (DCID_res, DB_commit_DBs, R)` ← *a* ∈*{yes, no}*] algorithm and finds out if there is any violation regarding user's data location preference and lets the user know about this. From the user's perspective, he/she will receive one of two messages as a response to their data location verification request; for example

"*Yes, your data is now in your preferred location, Europe*"
OR
"*Your data location preference has not been properly maintained*".

5 Discussion & Analysis

5.1 Security Analysis of DLAS

Here we do the security analysis of the DLAS solution for the attack scenarios considered in section 2.

Assumptions: Our security analysis is based on the following assumptions (which are already discussed in section 3).
Assumption 1: The CSP regularly (and correctly) sends the updated database (dataLocationDB_E) that contains data location information for all data of an Enterprise.

Assumption 2: The CSP has no control over the processes and databases) executed and managed respectively by the Enterprise, E, in the cloud.

Those assumptions are valid for the system model considered in Section 2. Assumption 1 makes sense since in this case the CSP sends data location information (basically just the pseudo-IDs of datacenter*s)* for all data, not for an individual data item; furthermore the database is sent offline and it

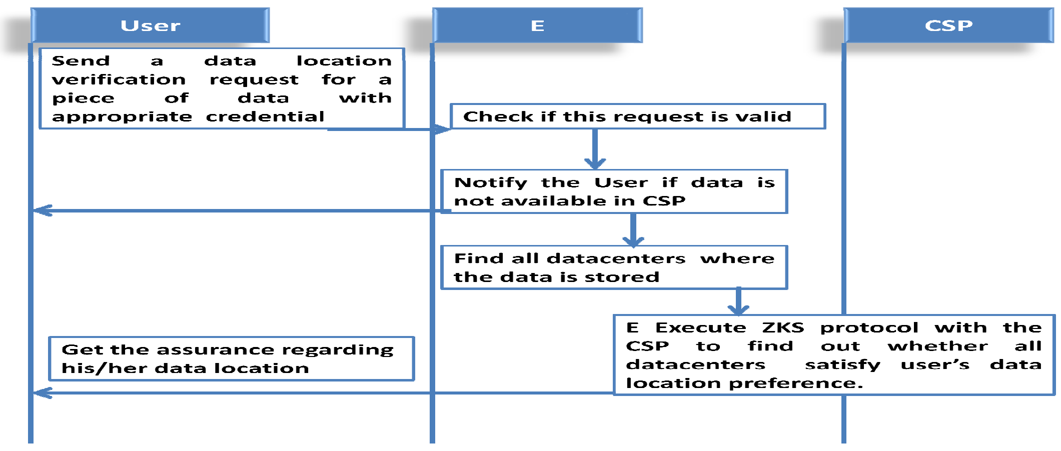
User
E
CSP
Send a data location verification request for a piece of data with appropriate credential
Check if this request is valid
Notify the User if data is not available in CSP
Find all datacenters where the data is stored
E Execute ZKS protocol with the CSP to find out whether all datacenters satisfy user's data location preference.
Get the assurance regarding his/her data location

Figure 7. The Data Location Verification Phase

is not sent as a response to some live request. Moreover the activity of regularly sending this database to the Enterprise should be a part of the SLA between the CSP and the Enterprise. Furthermore, assumption 2 is a practical example of IaaS (Infrastructure as a Service).

Case 1: Can the CSP lie about the data location for a particular data item: We make the following claim for this case:

CLAIM 1*: For the system and attack model considered for the DLAS solution, if the soundness property of the ZKS protocol holds, the CSP cannot lie about the location of the data (i.e. datacenter's location).*

Proof strategy: Claim 1 is of the form $A \rightarrow B$ where
A: the soundness property of the ZKS protocol holds
B: The CSP cannot lie about the location of the data
In order to prove claim 1, we will show that $\neg B \rightarrow \neg A$

Proof: This can be proved easily. First assume that the CSP can lie about the location of the data.

Now from assumption 1, the database *dataLocationDB_E* is correct. Furthermore the CSP cannot associate a specific user with specific data since it cannot get any information from the processes executed by the Enterprise in the cloud (based on assumption 2).

At this point, the CSP will **only** be able to lie about the location of the data by telling lies about the location of the datacenters. However the CSP has already committed all datacenters in the bootstrapping phase (which is assumed to be correct). So in order to be able to tell a lie about the location of the data, the CSP has to be able to generate both memebership and non-membership proofs for the same elements (i.e., for a dcID) in a given set (e.g., datacenter list in Europe). In other words, a CSP will be able to lie about the location of the data if it can show that:
(dcID $\in$ DCID) AND (dcID $\notin$ DCID)

Thus, if the CSP can lie about data location, then the ZKS protocol is not sound, and so assuming $\neg B$ implies $\neg$ A, which proves claim 1 ◊.

Note that Micali et al. has proved that the *soundness* property of the ZKS protocol holds (i.e., In the ZKS protocol, the probability for producing two proofs ($x \in S$) *and* ($x \notin S$) for the same element, x in a given set, S, by a prover and remain undetected to a verifier is very negligible); the proof can be found in [11].

Case 2: Enterprise's Knowledge about the CSP's datacenters: We make the following claim for this case:

CLAIM 2: *For the system and attack model considered for the DLAS solution, if the zero knowledge property of the ZKS protocol holds, the Enterprise cannot learn any additional information (regarding the datacenters of the CSP) other than just verifying the data location of a particular data item.*

Proof strategy: We follow the same proof strategy as for claim 1. We first assume that the Enterprise learns additional information regarding CSP's datacenter ($\neg B$) and then it causes the violation of zero knowledge property of the ZKS protocol ($\neg A$); in other words we show $\neg B \rightarrow \neg A$.

Proof: The proof is straight forward.
Assume that the Enterprise can learn additional information about the CSP's datacenter location. According to the system model, this is possible only if the Enterprise can extract more information (apart from checking the (non-) membership relation of a given set) from the committed database, *DB_commit*, which is only possible if the zero knowledge property of the ZKS protocol does not hold.

Informally '*zero knowledge*' means that an efficient Verifier learns only the values *D(x)* from his interaction with the Prover, and nothing else (not even the size of *D*). In particular the Verifier does not learn the values *D(x')* for an *x*' not queried to the Prover. The formal definition and proof of "zero knowledge" property of the ZKS protocol are given in [11]. In the context of the DLAS solution, it means that the Enterprise can know the location of at most *n* datacenters (i.e. *n dcIDs*) if all of its users' data is stored in *n* datacenters. Most importantly, the Enterprise will not be able to know even how many datacenters are located in a particular region (e.g. in USA). In addition, due to the adoption of the pseudo ids for the dcIDs, the Enterprise will see less repetition of *dcIDs*.

So we can conclude that $\neg$ B implies the violation of the *zero knowledge* property of the ZKS protocol (i.e., $\neg$ A) and thus claim 2 holds ◊.

5.2 Performance and Implementation Issues

Here we discuss performance and implementation issues for each phase of the proposed solution, DLAS, in order to justify the feasibility of adopting this DLAS solution in real cloud environments.

Bootstrapping Phase: This phase is off the critical path of the user's request. As a result, the performance (more precisely the timing issue) of this phase is not a significant issue for the DLAS solution.

Committing datacenters for each location preference list (i.e. coming up with *DB_commit* databases) using the ZKS protocol is the main operation of this phase. The commitment phase of the (original) ZKS protocol [11] is fast; in order to commit an elementary database, *DB*, it requires three modular exponentiations and one hashing for each element in the set (i.e. the EDB, *DB*). Note that in the DLAS solution, each element of an EDB represents a datacenter and in practice the number of datacenters in a particular region *R* (where *R* represents a country or a continent) would be in the range of 10-20 (at most). As a result, the EDBs used in the DLAS solution, will have a reasonable size.

Data Registration Phase: The performance of this phase is very important since it is on the critical path of a user's request.

The most expensive operation of this phase is the credential generation against a user's data storage request. Note that the credential is generated by executing the key generation algorithm of the CP-ABE algorithm. The key generation algorithm requires two exponentiations for every attribute assigned to the user, and the private key consists of two group elements for every attribute [12].

Offline Phase: This phase is off the critical path of a user request and thus its performance is not very crucial from the user's perspective. Recall that this phase only occurs once in a day.
The main goal of this phase is to build an encrypted database using the encryption algorithm of the CP-ABE scheme. The encryption algorithm requires two exponentiations for each leaf in the ciphertext's access tree and the ciphertext size will include two group elements for each tree leaf. As

noted earlier, the detailed description regarding the performance and implementation issues of the CP-ABE scheme is given in [12].

Data Location verification Phase: The performance of this phase is very crucial since it is on the critical path of a user's request.

This phase includes 3 main operations: decryption of the *dataLocationFinderDB* database (performed by the Enterprise), execution of the proof generation algorithm of the ZKS protocol (performed by the CSP) and the verification algorithm of the ZKS protocol (performed by the Enterprise).

According to [11], the decryption algorithm requires two pairings for every leaf of the access tree that is matched by a private key attribute and one exponentiation (at most 2) for each node along a path from such a leaf to the root. Note that in case of the DLAS solution, the access structure (i.e. access tree) would be very simple having a small number of nodes, which makes the decryption algorithm faster.

Between the proof generation algorithm and the verification algorithm, the proof generation algorithm dominates (in terms of computation and space requirements). In the proof generation algorithm of the ZKS protocol, the need for memory (not the computation) increases with the number of new proofs of *non-member*ship; fortunately the memory should not be an issue in case of the DLAS solution as the CSP will generate the proof and it has adequate computation and storage capability. Furthermore, it is also possible to adopt an improved ZKS protocol [25] which is able to generate shorter proofs than the original ZKS protocol; it can generate proofs of membership that are (approximately) 33% shorter, and proofs of non-membership that are almost 73% shorter than the original ZKS protocol.

[25] also includes a complete performance analysis of the original ZKS protocol for different security settings, it indicates that for a moderate size of EDB (having 200 elements with key of 120 bits), verification of a membership (or non-membership) proof requires less than 1 sec (and less than 5 seconds for non-membership) for a standard PC (e.g., Intel Core 2 Duo CPU at 2.4 GHz). Note that the EDBs of our DLAS solution should have less than 200 elements (which would represent datacenters in a given location) in real scenarios.

In one of our forthcoming papers, we will include a prototype implementation of the DLAS solution in order to validate these performance numbers.

5.3 Further Comments on the Proposed Solution:

Why did we choose CP-ABE? In the proposed DLAS solution, it might be possible to consider other means of access control solutions instead of the CP-ABE scheme. However one of the main reasons for choosing CP-ABE is that it is very generic and flexible as well. On one hand, it can be used to develop the typical encryption scheme which allows only a valid user to decrypt; on the other hand, it can also be used in the RBAC (Role Based Access Control) model where a user's access will depend on its role. Another significant advantage of CP-ABE is that you can even put the encrypted database (using CP-ABE) into a malicious server. Last, but not least, we also consider the fact that the implementation of the CP-ABE scheme is publicly available [24].

Room for improvements in the ZKS protocol: The ZKS (Zero Knowledge Set) protocol is one of the main building blocks of our proposed DLAS solution. At present there exist a number of papers [25, 26] which have proposed some sort of improvements (in terms of performance or security) over the original ZKS protocol proposed by Micali et al. in [11]. However we considered the original ZKS protocol for our DLAS solution since it is comparatively simple; furthermore, in this paper our intention is just to present a basic construction of DLAS solution, and for this, the original ZKS protocol is more suitable than [25, 26].

Further comments on the assurance received from the CSP: Note that the verifiable assurance a user receives in the DLAS solution reflects (at most) 24 hours' earlier state of their data location since the database *dataLocationDB_E* is sent to the Enterprise once in a day (at 12:00 am everyday). However, the CSP cannot lie to the users about that state.

6 Conclusion

In this paper, we have proposed the DLAS, the first non-geolocation based data location assurance solution for cloud storage environments. The system model we considered (the 3-party model) for our DLAS solution is very common in real cloud environment settings; furthermore our attack model ('honest-but-curious server model') is also very well-known and widely used in many security solutions for cloud environments. The main reason behind coming up with a non-geolocation based solution (instead of a geolocation based solution) is that "geo-location" based solution cannot fulfill one of the important needs of existing cloud storage environments, which is 'geo-replication' of data; more precisely, existing geolocation technique (e.g.; ping) based data location assurance solutions are unable to provide data location assurance to the users when a piece of data is stored/copied in multiple datacenters/servers.

Our DLAS solution allows users to give their data location preference in the following way: "Keep/ do not keep my data in *region R* (e.g., *R* can be a country or continent)"; later, users will receive *provable* assurance of their data location from the CSP (Cloud Storage Provider) with respect to their data location verification request. We also include a security and performance analysis of the DLAS which shows the feasibility of adopting the DLAS solution for real cloud environments. Unlike other existing solutions [8, 9], the DLAS solution also satisfies the geo-replication practice (i.e., a piece of data can be stored in multiple datacenters) of the existing CSPs (e.g., Windows Azure).We also believe that DLAS has the potential to be used as one of the building blocks of the "accountable cloud" [2].

7 Future Work

For one of our forthcoming papers, we are implementing a prototype of our proposed solution, DLAS, to confirm and validate the theoretical performance numbers given in this paper. We are also designing a DLAS solution for a different system model (having only 2 party instead of 3) where a cloud user will directly interact with the CSP (i.e., without the intervention of Enterprise). The current DLAS solution is based on an "honest-but-curious" cloud server model; as future work, we are also considering the possibility of a data location assurance solution for the "dishonest" cloud server attack model, which is another important and challenging attack model.

REFERENCES

[1] Anthes, Gary. "Security in the cloud." *Communications of the ACM* 53.11 (2010): 16-18..

[2] Haeberlen, Andreas. "A case for the accountable cloud." *ACM SIGOPS Operating Systems Review* 44.2 (2010): 52-57.

[3] ITCandor, Retrieved April 15, 2013 from
http://www.itcandor.com/usa-cloud-2012

[4] Chen, Yanpei, Vern Paxson, and Randy H. Katz. "What's new about cloud computing security?." *University of California, Berkeley Report No. UCB/EECS-2010-5 January* 20.2010 (2010): 2010-5.

[5] Chaves, Shirlei, et al. "Customer Security Concerns in Cloud Computing." *ICN 2011, The Tenth International Conference on Networks*. 2011.

[6] Heiser, Jay, and Mark Nicolett. "Assessing the security risks of cloud computing." *Gartner Report* (2008).

[7] Noman, Ali, and Carlisle Adams. "DLAS: Data Location Assurance Service for cloud computing environments." *Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on*. IEEE, 2012.

[8] Albeshri, Aiiad, Colin Boyd, and Juan Gonzalez Nieto. "GeoProof: Proofs of Geographic Location for Cloud Computing Environment." *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*. IEEE, 2012..

[9] Watson, Gaven J., et al. "LoSt: location based storage." *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*. ACM, 2012.

[10] Peterson, Zachary NJ, Mark Gondree, and Robert Beverly. "A position paper on data sovereignty: The importance of geolocating data in the cloud." *Proceedings of the 8th USENIX conference on Networked systems design and implementation*. 2011.

[11] Micali, Silvio, Michael Rabin, and Joe Kilian. "Zero-knowledge sets."*Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*. IEEE, 2003.

[12] Bethencourt, John, Amit Sahai, and Brent Waters. "Ciphertext-policy attribute-based encryption." *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007.

[13] Di Vimercati, Sabrina De Capitani, et al. "Over-encryption: management of access control evolution on outsourced data." *Proceedings of the 33rd international conference on Very large data bases*. VLDB endowment, 2007..

[14] Yu, Shucheng, et al. "Achieving secure, scalable, and fine-grained data access control in cloud computing." *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010.

[15] Li, Jin, et al. "Fine-grained data access control systems with user accountability in cloud computing." *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010.

[16] Li, Ming, et al. "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings." *Security and Privacy in Communication Networks* (2010): 89-106.

[17] Wang, Guojun, Qin Liu, and Jie Wu. "Achieving fine-grained access control for secure data sharing on cloud servers." *Concurrency and Computation: Practice and Experience* 23.12 (2011): 1443-1464.

[18] Popovic, Kresimir, and Zeljko Hocenski. "Cloud computing security issues and challenges." *MIPRO, 2010 Proceedings of the 33rd International Convention*. IEEE, 2010.

[19] Gill, Phillipa, et al. "Dude, where's that IP?: circumventing measurement-based IP geolocation." *Proceedings of the 19th USENIX conference on Security*. USENIX Association, 2010.

[20] Wang, Qian, et al. "Enabling public verifiability and data dynamics for storage security in cloud computing." *Computer Security–ESORICS 2009* (2009): 355-370.

[21] Chow, Richard, et al. "Controlling data in the cloud: outsourcing computation without outsourcing control." *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009.

[22] Mell, Peter, and Timothy Grance. "The NIST definition of cloud computing (draft)." *NIST special publication* 800 (2011): 145.

[23] Calder, Brad, et al. "Windows Azure Storage: a highly available cloud storage service with strong consistency." *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011.

[24] Advanced Crypto Software Collection, Retrieved March 31, 2012 from
http://acsc.cs.utexas.edu/cpabe/

[25] Catalano, Dario, Dario Fiore, and Mariagrazia Messina. "Zero-knowledge sets with short proofs." *Advances in Cryptology–EUROCRYPT 2008* (2008): 433-450.

[26] Chase, Melissa, et al. "Mercurial commitments with applications to zero-knowledge sets." *Advances in Cryptology–EUROCRYPT 2005* (2005): 605-605.

[27] Gondree, Mark, and Zachary NJ Peterson. "Geolocation of data in the cloud." *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013.