"One of Our Hosts in Another Country": Challenges of Data Geolocation in Cloud Storage

Nicolae Paladi
Security Lab
Swedish Institute of Computer Science
& Lund University, Sweden
nicolae@sics.se

Antonis Michalas
Security Lab
Swedish Institute of Computer Science
Stockholm, Sweden
antonis@sics.se

***Abstract*—Physical location of data in cloud storage is an increasingly urgent problem. In a short time, it has evolved from the concern of a few regulated businesses to an important consideration for many cloud storage users. One of the characteristics of cloud storage is fluid transfer of data both within and among the data centres of a cloud provider. However, this has weakened the guarantees with respect to control over data replicas, protection of data in transit and physical location of data. This paper addresses the lack of reliable solutions for data placement control in cloud storage systems. We analyse the currently available solutions and identify their shortcomings. Furthermore, we describe a high-level architecture for a trusted, geolocation-based mechanism for data placement control in distributed cloud storage systems, which are the basis of an on-going work to define the detailed protocol and a prototype of such a solution. This mechanism aims to provide granular control over the capabilities of tenants to access data placed on geographically dispersed storage units comprising the cloud storage.**

***Index Terms*—Trusted Cloud Geolocation, Secure Cloud Computing, Data Protection**

I. Introduction

In the last few years, cloud computing has seen a rapid evolution and tends to be one of the fastest growing segments of the IT industry. Cloud computing offers additional benefits to businesses by providing flexible infrastructure at lower costs, greater scalability and improved disaster recovery.

As a result, the amount of data stored in the cloud by individuals and companies has grown at a rapid pace. However, despite strong trends towards migration of workloads and storage to the clouds, many users and decision makers voice concerns about the security and privacy of their data. In the early stages of cloud computing, users were sceptical of the capabilities of cloud computing to securely store their data. This increasing demand for security mechanisms to ensure the safety of stored data resulted in extensive research efforts to build protocols for secure storage protection [1]–[3]. However, the continuous evolution of services offered by cloud providers are expanding demonstrated that storage protection mechanisms alone seem insufficient.

Lately, along with the already traditional questions about the safety of cloud environments we have also seen concerns about the physical location of data and its availability in different jurisdictions. By storing data in the cloud, users hand it over to a provider that may have data centres in different geographical locations, countries or even continents. However, organizations that work with sensitive data – such as health records, or financial data – need complete control over the physical storage location and data access. As a result, storing sensitive data in the cloud complicates adherence to regulatory compliance laws, since such data may fall under different regulations depending on where it is physically stored. If for example data is moved to a different country, a different set of rules may apply.

Data location is one of the most common compliance issues that an organization faces [4], [5]. Usage policies and options for physical placement of data in the cloud will impact the decision of an organization with regard to usage of cloud services. By moving data beyond their technological and geographic borders, organizations risk losing control of how their data complies with the local legal frameworks. There is thus a clear need for mechanisms that would allow all cloud storage users to select the territories where their data can be stored as well as verify the exact location of their data.

A. Our Contribution

In this invited paper, we present a theoretical analysis of the existing trusted geolocation systems for the cloud. We use this analysis to demonstrate the inefficiencies, as well as the strengths of existing protocols and redefine the important research questions in cloud data geolocation in the hope to spawn further research in the area. To the best of our knowledge, this is the first work that provides a collective description of such systems. This description can provide essential knowledge to protocol designers and help them avoid common pitfalls and design better trusted geolocation solutions for the cloud storage.

B. Organaziation

In Section II we define the problem of trusted geolocation in cloud computing and the primitives used throughout the paper. In Sections III and IV, we analyze the main proposed solutions for data geolocation in cloud storage. In Section V, we present a novel high-level architecture for data placement control in cloud storage while in Section VI we conclude the paper.

II. Problem Statement & Definitions

In this section, we define the problem of trusted geolocation in cloud computing along with the primitives that we use in the rest of the paper. All of the protocols presented in this paper consider a cloud service provider (CSP) which uses a set of hosts distributed in different geographic locations.

CSP's Locations & Hosts: We assume that a CSP uses a set of geographically distributed hosts. Let $L = \{l_1, \ldots, l_n\}$ be the set of all locations where CSP can store user data. Then the set $S_i = \{s_1^i, \ldots, s_k^i\}$ 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 $T_i \subseteq L$ be the set of all trusted locations for user u_i. Then the set of hosts in T_i is denoted as $S_{T_i} = \{s_1^{T_i}, \ldots, s_l^{T_i}\}$.

Distance Between a Host & a Location: Most of the protocols presented in this paper 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:

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

Proof of Retrievability (PoR): PoR was introduced in [6] 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 $POR(P, V)$ an execution of the protocol between a prover P and a verifier V.

Trusted Platform Module (TPM): TPM is a tamper-evident hardware cryptographic coprocessor which follows the specifications of the Trusted Computing Group [7]. An active TPM records the software state of the platform at boot time and stores it in its platform configuration registers (PCRs) as a list of hashes.

Problem Statement: *Let $U = \{u_1, \ldots, u_n\}$ 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 of locations in which f should be stored;*
2) *u_i must have the option to validate the location of f at any time;*
3) *u_i must ensure that CSP does not store plaintext replicas (f_{ri}) of f outside of the allowed geographical area(s);*

III. Distance Based Protocols

Despite the abundance of secure storage and trust establishment schemes in cloud computing, there is a clear lack of protocols that address the problem of data placement control. Furthermore, most of the existing work relies on techniques where the location of a host is calculated based on the round trip time (RTT) measurements between a responder (host) and a requester (user, CSP or a third party). In the remainder of this section, we will analyze the most important trusted geolocation schemes published to date. This analysis will help us expose drawbacks and inefficiencies of the existing works as well as the strengths of each protocol.

Before proceeding with the description of the protocols, we need to describe their adversarial model. Most of the following protocols assume an economically rational adversary that aims to reduce costs through data migration in spite of contractual agreements. As described in [8] and [9] an economically rational adversary will avoid storing data needlessly but can deviate from the protocol when it believes it is being audited.

A. LoSt: Location Based Storage [10]

Watson *et al.* [10] showed that there are limits to the accuracy of verifying the location of data in a cloud storage. The authors demonstrated that when a malicious CSP colludes with malicious hosts, it is infeasible for a user to correctly verify the exact location of the files. Furthermore, authors of [10] were the first to take into consideration cases where two or more malicious hosts collude and make copies of the stored files. This assumption led them to argue that the task of restricting where the geographic location of data is impossible. In addition, they suggested a proof of location (PoL) scheme that can be used by a user in order to obtain the location of a stored file. The proposed protocol is composed of two major phases, *Store* and *Locate*.

Store: A user u_i wishes to store a file f in a certain location $l_j \in T_i$. To do this, u_i encodes f, generates a tag tag_{l_j} and sends it together with l_j and the encoding type to CSP. Upon reception, CSP is responsible to select a set of hosts ($\mathcal{T} \subseteq S_i$) within l_j that will store the encoded file $E(f)$ as well as replicas of the file in each host. Additionally, CSP can re-encode each replica with a different key for each host.

Locate: In this phase u_i needs to verify that $E(f)$, as well as all its copies, are stored on hosts within the geographical borders of l_j. During the initialization step, u_i sends a request to CSP querying the list $\mathcal{T}$ of all hosts which store f. At this point, authors assume that there is a trusted landmark infrastructure $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$, independent from the storage system, through which a user can interact with the hosts of the CSP. u_i forwards S_f to the trusted landmarks that are responsible for individually challenging each host in the list. Each landmark $\mathcal{L}_i$ verifies that each host in S_f holds a copy of f by running $POR(\mathcal{T}, \mathcal{L}_i)$, $\forall\ s \in \mathcal{T}$. Then, for every host that proves the fact of holding a copy of f, $\mathcal{L}_i$ calculates the approximative location by measuring the network latency. The results are then returned to u_i, which is responsible for verifying that each location is an element of the set T_i.

Strengths: The main advantage of LoSt is the analysis which yield list of attacks applicable to protocols that use latency based techniques in order to locate data stored in the cloud. More precisely, authors presented two attacks that allows a malicious server to lie about its location. The first attack is called *shortening*, where a malicious server $\mathcal{ADV}$ pretends to be closer to a location than it really is; the second one is called *lengthening*, where $\mathcal{ADV}$ pretends to be farther from a location. Every time $\mathcal{ADV}$ receives a request to prove that holds a file f, it uses the $dist$ function in order to calculate the distance of its current location with the claimed location. If the distance is smaller than the one that verifier is expecting, $\mathcal{ADV}$ simply adds the appropriate delay to the communication. In the case where the distance is greater than the claimed one, $\mathcal{ADV}$ colludes with a proxy server which is close to the claimed location and holds f. The proxy will be responsible for

answering the challenges needed for the verification on behalf of $\mathcal{ADV}$.

Furthermore, authors showed how to construct a PoL scheme from a geolocation and a PoR scheme. Additionally, they improved the efficiency of the construction by introducing recoding, a new property for PoR's which reduces the computation and communication cost of the user.

Limitations: To verify of the location of a stored file, authors posit that a set of trusted landmarks exists and will be responsible for verifying the existence of a file on a host. This requirement makes the protocol difficult to implement in practice, since it would require major changes to the infrastructure of a cloud provider.

B. GeoProof [11]

A. Albeshri *et al.* [11], [12] 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 CSP and a third party will communicate with this device in order to verify the location of the stored data on behalf of a user.

Similar to the previous protocol, GeoProof consists of two phases, *Store* and *Locate*.

Store: During the initialization step, u_i splits a file f into n shares $f = \{f_1, \ldots, f_n\}$. From the calculated shares, u_i creates k-block chunks and applies an error correction code to each chunk. This process results in the generation of a new file f', where u_i encrypts the data with a symmetric algorithm, reorders the blocks of $E(f')$ and splits the result into n shares $E(f') = \{E_{f_1}, \ldots, E_{f_n}\}$. For each generated share, u_i calculates the MAC value and the result is the file f_{final} that will be stored in the cloud.

Locate: In order for u_i to verify the location of a file, she sends f_{final} to the CSP. Recall that f_{final} consists of n shares and each share is associated with a tag tag_i. The third party generates a random nonce r and sends it back to u_i along with the total number of shares that f_{final} contains as well as the number of shares that will be checked during the verification protocol. Upon reception, u_i generates a random challenge c, such that $c = \{c_1, \ldots, c_k\} \subseteq \{1, \ldots, n\}$. Then, for each c_i, u_i sends c_i to the CSP. Upon reception, CSP finds the corresponding block from f_{final} and replies to u_i by sending the corresponding share concatenated with the corresponding tag. After k rounds, u_i will have received responses for all the challenges from the set c. u_i can then find the approximative physical location of the file by calculating the RTT.

Strengths: In [12] authors present an enhancement of the original paper [11] where they manage to avoid the delay in computation of the PoR scheme and thus reduce the host-side delay in returning the PoR proof. Furthermore, storage protection is provided by both encrypting the file to be stored and by splitting and creating a random order for the generated shares, thus hindering the attacker from reconstructing the file. Finally, authors provided experimental results for *hard disk*, *LAN* and *Internet* latency.

Limitations: The GeoProof protocol suffers from two main drawbacks. First, the paper does not address the *copies* of the data stored in remote locations. Second, even though distance-bounding protocols are very sensitive to timing delays, no solution is proposed for such attacks. Furthermore, there is no discussion on how a user can select the locations in which she wishes to store a file. Finally, the paper does not provide any solution to uniquely identify the storage hosts of the cloud service provider.

C. Do You Know Where Your Cloud Files Are? [8]

Authors address the question of determining the physical location of data in a distributed IaaS storage. Benson *et al.* propose a method for determining the location of data in IaaS storage with a per-data center granularity. The solution assumes that the locations of all data centres where the CSP stores data are known, that the CSP 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 uses the Haversine distance[1] 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.

In addition, 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.

Strengths: The protocol does not require additional reference points, except the known locations of data centres. Furthermore, authors provide extensive experimental results by collecting data from different locations using the PlanetLab test bed [13].

Limitations: The main drawback of the proposed solution is that it does not deal with *copies* of the data stored in remote locations. Ignoring this aspect makes the problem of trusted geolocation trivial. Moreover, authors incorrectly assume that the CSP does not have dedicated communication channels between the data centres, ignoring the use of *dark fiber* by CSPs. Finally, during the experiments the authors do not include the measurements regarding packet loss, making the results unreliable since a malicious provider could always drop packets.

D. Geolocation of Data in the Cloud [9]

Gondree and Peterson [9] proposed a Constraints-Based Data geolocation (CBDG) solution for determining the location of data and its "binding" to specific locations[2]. More precisely, they extend the solutions in [8], [14] by providing a generic framework for actively monitoring the location of stored data in the cloud using latency based techniques.

The suggested approach combines probabilistic provable data possession with geolocation in a CBDG protocol, which builds closely on a MAC-based PoR scheme with some additional steps. The protocol assumes an initial model building stage,

[1]The Haversine formula (haversin(θ) = $sin^2(\frac{\theta}{2}) = \frac{1-cos(\theta)}{2}$) is used to calculate great-circle distances between two points on a sphere from their longitudes and latitudes.

[2]Binding is here used in the sense of detecting occurrences of data misplacement, rather than data binding in the meaning common in trusted computing

where a set of landmarks $\mathcal{L}$ throughout the analyzed geographical region, each building a latency-distance estimation model. Furthermore, PoR challenges (similar to the challenge queries $\mathcal{Q}$) in [15] are divided between $\mathcal{L}$. Using its latency-distance model, each landmark generates a circular constraint of a radius $r_{\mathcal{L}}$ centered on $\mathcal{L}$. The geolocation step of the protocol uses the intersection of geolocation constraints $[r_{\mathcal{L}}]$ to determine the region where the data resides.

Strengths: The proposed solution combines provable data possession with data geolocation. In addition to that, it provides a more generic framework than the previous works ([8], [14]), which allows any distance latency model – including topology-aware models – to be used.

Limitations: Despite the advantages over the previous works [8], [14], the proposed solution does not overcome their main drawbacks. It does not provide any solution for replicas of the data stored in remote locations; furthermore, proper functioning of the protocol requires a set of landmarks placed close to the data centres of the CSP, in order to run latency-based distance estimation models.

IV. Using a Hardware Root of Trust in Data Geolocation

In Section III, we reviewed several proposed protocols for data geolocation in cloud storage and exposed a range of critical limitations of the distance-bounding protocols applied to data geolocation. A complementary approach is to store geolocation data directly on the storage host, protected by a hardware root of trust. We continue by a review of two proposed solutions that rely on a hardware root of trust in order to either control or verify the placement of data in a particular jurisdiction [16], [17].

A. Leading the way: NIST 7904

The National Institute of Standards and Technology (NIST) has described a prototype implementation for trusted geolocation in the cloud [17]. The prototype relies on the combination of trusted computing, Intel Trusted Execution Technology and a set of manual audit steps in order to verify and enforce data location policies. This is done in several stages: *(i)* platform attestation and safe hypervisor launch, which ensures that the host platform is running a trusted software stack; this stage results in the creation of a *trusted computing pool* composed of hosts with an attested platform state; *(ii)* out-of-band provisioning of geolocation data to the platforms in the trusted computing pool; *(iii)* trust-based and geolocation-based workload migration – this stage aims to ensure that workloads (such as data storage and computation) are only placed on hosts that belong to the trusted computing pool.

The hardware root of trust is defined as "inherently trusted combination of hardware and firmware that maintains the integrity of the geolocation information and the platform.", e.g. TPM. The solution assumes that geolocation information is bound along with platform metadata and stored in the TPM. The information is later accessed in stage *(iii)* in order to verify the integrity of the host and the location of the platform.

Strengths: The use of a hardware root of trust provides some additional guarantees with regard to the trustworthiness of the platform in the protocol context.

Limitations: The described proof of concept is based on a closed-source software stack; the solution description lacks important elements and does not explain the mechanism of the out-of-bound provisioning of geolocation information to the non-volatile index of the TPM; finally, the solution relies on policy assurances but does not describe a strong link between the declared geolocation of the platform and placement of the workload. The lack of reliable assurance mechanisms increases the risks brought by trivial operator mistakes and policy misconfiguration which could result in migrating clear text data outside the trusted computing pools.

Given the fact that the prototype described in [17] is built using proprietary software, we can not present a practical attack against this approach. However, we can perform a theoretical vulnerability assessment. While [17] does not describe a specific mechanism for out-of-band provisioning of geolocation data, a theoretical attack would be to *impersonate* the authority provisioning the geolocation tag. The goal is to provision false geolocation information (using the same unnamed out-of-band mechanism) to a host with trusted configuration s_1^i, placed in an illegal location i. Since in this case the TPM is a passive consumer of geolocation data, it would extend it to the respective PCR in the TPM during the platform launch. As a result, the host s_1^i would present geolocation information corresponding to a host s_1^T and would therefore be eligible to access restricted data and workloads.

Storing information location with the CA

The authors of [16] propose a different approach regarding the use of TPMs on host platforms for data geolocation in clouds. The solution assumes that the identity of the host's TPM is stored along with the host's geographical position with the Certificate Authority (CA). Next, the user of a VM instance would request an attestation of the host platform in order to obtain trust guarantees and geolocation data. The solution further assumes that all VM instances implement a "LocCheck" client which is able to communicate with a "Location verification and integrity check" (LICT) module implemented in the hypervisor. A two-stage protocol is suggested: the *initialization phase* includes remote attestation of the host platform and *physical* verification of host location; and the *verification phase*, where the user randomly chooses several memory areas (MA) $MA_1, \ldots, MA_n$ and interacts with the VM's "LocCheck" client that communicates with the virtualization host to obtain all the necessary manipulations with the local TPM to verify (through a mediation of the CA) that data is indeed stored on the platform.

Strengths: Similar to the approach in IV-A, the use of a hardware root of trust provides some additional guarantees with regard to the trustworthiness of the platform in the context of the protocol.

Limitations: The solution assumes that geolocation verification is done through administrative methods. Besides being costly and often infeasible due to the sheer number of platforms, this approach does not prevent data replicas from being stored in other, arbitrary locations. Furthermore, the paper does not describe any implementation results.

While the authors of [16] do not present a detailed implementation design, we can present a simple theoretical relay attack

for this model. Since in the presence of a malicious CSP the user u_i can not know which VM instance they communicate with, the CSP can induce u_i to believe they are communicating with an instance VM^A running on a host in a trusted location, s_1^T, when in fact it communicates with a modified instance VM^B located on a host in an illegal location, s_1^I. When u_i decides to verify the location of the instance it communicates with, it chooses – according to the protocol – a range of memory values $MA_1, \ldots, MA_n$. The memory values are then copied on to a VM^A which is launched on s_1^T, which relays the rest of the communication according to the protocol. While the authors discuss the lack of incentive for a form of a relay attack from the CSP because it would require always running two copies of VM instances, the above described relay attack only requires the instance VM^A to be launched on s_1^T during the geolocaton verification according to the protocol. Since the verification according to the protocol is not likely to be done continuously – otherwise the u_i would themselves cause a denial-of-service of the TPM on the host, given that the TPM has very limited processing power – periodically launching VM^A on s_1^T would make sense from an attacker's point of view.

V. Trusted geolocation-based data placement

Based on the analysis of the currently proposed solutions for data geolocation in cloud storage, we provide a new conceptual architecture that overcomes the above limitations. While a detailed description and implementation of this architecture will be presented in an upcoming report, we present here a set of requirements that such an architecture should satisfy, along with a high-level overview of the protocol. We consider a malicious CSP that provides a storage service (using a large-scale distributed data store) and attempts to store plain text copies of user-provided data in arbitrary jurisdictions, ignoring the requirements of its clients; however, we assume the physical security of storage hosts. The proposed solution aims to fulfil the following requirements:

1) Plain text data – including all plaintext copies and replicas – may only reside on storage hosts in $\mathcal{T}$;
2) Requirement 1 must hold in the face of both accidental policy misconfigurations and deliberate attempts of the CSP to place plaintext data on hosts outside $\mathcal{T}$
3) A user u_i should not need to verify post factum that her data is placed in the chosed jurisdiction $\mathcal{J}_i$;
4) A third party – e.g. an auditor – should be able to uniquely identify the specific hosts where plain text data (including copies) may reside;

We make several assumptions for the purposes of the solution: *(i)* physical security of the data centres is ensured; *(ii)* the server platforms are equipped with a hardware root of trust, e.g. TPM; *(iii)* all server platforms are equipped with a GPS receiver. Assumptions *(i),(ii)* are trivial; however, a remark is necessary in the case of assumption *(iii)*. While GPS receivers are currently common on mobile and laptop platforms, they are less common on server platforms. However, given that the hardware already exists, we can make a parallel with the evolution of the TPM (which was initially deployed on laptops) to assume that GPS devices will shortly be widely available for server platforms as well.

The solution assumes that a u_i uploads exclusively encrypted data to the cloud storage. The decryption key – along with a hash of the data and the requirement to store the data in jurisdiction $\mathcal{J}_i$ – is protected with the public key of the TTP and uploaded along the data. Storage hosts record their platform state in TPM registers at boot time, as well as the location data read from the GPS receiver by a kernel module that communicates between the drivers of the GPS receiver and the TPM. Upon request from the CSP, TTP attests the storage hosts of the CSP that are placed within $\mathcal{J}_i$ and *seals* the decryption key to the trusted configuration of the platform and the location of the platform. Finally, the storage hosts decrypt the data provided by u_i for further processing.

VI. Conclusions

In this invited paper, we have presented an analysis of protocols aiming to provide geographic location assurance for cloud computing environments, as well as provided a thorough list of strengths and limitations. We hope this comprehensive list of protocol inefficiencies provide essential knowledge to protocol designers and will inspire further research in the area to design better trusted geolocation solutions for cloud storage. Based on the identified limitations, we have presented a high-level architectural description of an alternative, trusted geolocation-based data placement mechanism for cloud storage systems. This trusted geolocation-based data placement mechanism is currently under development and will be presented in future work. Considering that this field is quite young, more work is needed to identify geolocation mechanisms with stronger security guarantees and a minimal impact on the functionality of cloud storage systems.

References

[1] N. Paladi, C. Gehrmann, and F. Morenius, "Domain-Based Storage Protection (DBSP) in Public Infrastructure Clouds," in *Secure IT Systems*, pp. 279–296, Springer, 2013.

[2] M. Rezaei, N. Moosavi, H. Nemati, and R. Azmi, "Tcvisor: A hypervisor level secure storage," in *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*, pp. 1–9, IEEE, 2010.

[3] S. Graf, P. Lang, S. Hohenadel, and M. Waldvogel, "Versatile key management for secure cloud storage," *Submitted at EuroSys*, vol. 11, no. 11.4, pp. 2012–13, 2012.

[4] W. A. Jansen, "Cloud hooks: Security and privacy issues in cloud computing," in *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, HICSS '11, (Washington, DC, USA), pp. 1–10, IEEE Computer Society, 2011.

[5] B. R. Kandukuri, R. P. V., and A. Rakshit, "Cloud security issues," in *Proceedings of the 2009 IEEE International Conference on Services Computing*, SCC '09, (Washington, DC, USA), pp. 517–520, IEEE Computer Society, 2009.

[6] 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.

[7] Trusted Computing Group, "TCG Specification, Architecture Overview, revision 1.4," tech. rep., Trusted Computing Group, 2007.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] L. Peterson, S. Muir, T. Roscoe, and A. Klingaman, "PlanetLab Architecture: An Overview," Tech. Rep. PDN–06–031, PlanetLab Consortium, May 2006.

[14] 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.

[15] 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.

[16] C. Krauß and V. Fusenig, "Using trusted platform modules for location assurance in cloud networking," in *Network and System Security*, pp. 109–121, Springer, 2013.

[17] E. Banks, M. Bartock, K. Firftal, D. Lemon, K. Scarfone, U. Shetty, M. Souppaya, T. Williams, and R. Yeluri, "DRAFT Trusted Geolocation in the Cloud: Proof of Concept Implementation," *NIST special publication*, vol. 7904, p. 42, 2012.