Assured Deletion in the Cloud: Requirements, Challenges and Future Directions

Kopo M. Ramokapane
Security Lancaster Research Centre
Lancaster, United Kingdom
k.ramokapane@lancaster.ac.uk

Awais Rashid
Security Lancaster Research Centre
Lancaster, United Kingdom
a.rashid@lancaster.ac.uk

Jose M. Such
Security Lancaster Research Centre
Lancaster, United Kingdom
j.such@lancaster.ac.uk

ABSTRACT

Inadvertent exposure of sensitive data is a major concern for potential cloud customers. Much focus has been on other data leakage vectors, such as side channel attacks, while issues of data disposal and assured deletion have not received enough attention to date. However, data that is not properly destroyed may lead to unintended disclosures, in turn, resulting in heavy financial penalties and reputational damage. In non-cloud contexts, issues of incomplete deletion are well understood. To the best of our knowledge, to date, there has been no systematic analysis of assured deletion challenges in public clouds.

In this paper, we aim to address this gap by analysing assured deletion requirements for the cloud, identifying cloud features that pose a threat to assured deletion, and describing various assured deletion challenges. Based on this discussion, we identify future challenges for research in this area and propose an initial assured deletion architecture for cloud settings. Altogether, our work offers a systematization of requirements and challenges of assured deletion in the cloud, and a well-founded reference point for future research in developing new solutions to assured deletion.

Keywords

Assured deletion, Secure deletion, Public cloud computing, Cloud computing security, User assurances

1. INTRODUCTION

For cloud computing customers, disclosure of sensitive data is a major concern. While research has investigated side channel attacks and various other types of attacks exploiting virtualization [23, 26, 45], security issues arising from insecure or incomplete data deletion have not been considered. However, incomplete data deletion may lead to unintentional or premediated exposure of tenants' sensitive information. The costs associated with such disclosures are high and include financial losses (for both customers and providers, e.g., through regulatory fines) and loss of reputation. For instance, [29] reported that a bug in a provider's application programming interface (API) allowed new tenants to read data belonging to former tenants. The provider's system failed to scrub the disks before reallocating them to new tenants.

However, it is important to note that assured deletion guarantees in the cloud are not only important to tenants, they are also important to providers. From a tenant perspective, it is essential to receive assurances that data will be handled and destroyed as agreed. From a cloud provider perspective, such guarantees are needed to comply with data regulations of various countries and regions and also address tenants' requirements and expectations. Furthermore, deletion assurances can also become a differentiator in the market for a cloud provider.

Significant efforts have been made in various areas to provide cloud users with assurances such as proof of data availability [8, 24, 25], data integrity [8, 47], data location [8, 52] and encryption [16]. But, for deletion, cloud tenants are made to rely heavily on trust – trusting that the provider will ensure that data is completely deleted upon request. Sometimes such assurances are included in the contracts and service level agreements (SLAs) but they still require trusting the provider without any technical proof [20]. Technical assurances – and proof of deletion upon request – can give tenants confidence about how their outsourced data is handled and decommissioned.

In non-cloud contexts issues of insecure deletion are well understood [12, 41, 42, 44]. For cloud tenants, having no access to the infrastructure makes it difficult for them to verify deletion of data from the cloud while, for providers, there are a number of salient features such as service delivery models, multi-tenancy, virtualization, elasticity, high availability and data backup, all of which pose various challenges with regards to providing assured deletion guarantees. Current research work in this area has mainly focused on scenarios where the tenant wants assured deletion when using an untrusted cloud provider [49, 40, 27]. Moreover, such works focus only on cloud storage providers, neglecting applications involving data processing in the cloud or scenarios where the cloud provider may be interested in providing deletion assurances.

To contextualize the challenges of assured deletion in the cloud, we present two adversarial models, one involving a *dishonest cloud provider* and the other an *honest cloud provider*. Using the *dishonest cloud provider* scenario, we draw and present requirements for assured deletion for a tenant, and

CCSW'16, October 28 2016, Vienna, Austria

ISBN 978-1-4503-4572-9/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2996429.2996434

then analyse existing solutions for assuring deletion in such scenarios. We discuss the limitations of such solutions and the open research challenges. Afterwards, using the *honest provider model* we draw requirements for assured deletion in such a context. In the absence of existing solutions in extant literature, we review OpenStack, an existing cloud infrastructure. We analyse, in depth, the OpenStack feature set and the challenges the various features pose with regards to assured deletion. In both adversarial scenarios, we identify both open research problems and current areas of research that might act as building blocks or help towards achieving assured deletion in the cloud.

In summary, our main contributions are: as follows

- We discuss the issue of assured deletion in the cloud from two perspectives, the dishonest provider's point of view and the honest provider's point of view, providing a distinctive mapping between requirements and challenges.
- We identify critical cloud features which pose challenges to assured deletion, and offer a systematic analysis and discussion of these challenges for assured deletion for both cloud tenants and providers.
- We discuss open challenges for assured deletion in the cloud, for both the tenant and the provider.

As a result, our work is significant with many benefits. For researchers, it offers an initial study of the subject; for practitioners, it provides a comprehensive study of the existing solutions and identifying limitations and challenges in the area.

The remainder of this paper is organized as follows. Section 2 introduces assured deletion and presents the two adversarial models discussed in this paper. Section 3 discusses the dishonest cloud provider model by first presenting assured deletion requirements, and then providing an analysis of existing solutions against the requirements. It concludes by presenting the limitations of the existing solutions and possible future research areas. Section 4 starts by presenting assured deletion requirements for the honest cloud provider model followed by an analysis of current cloud infrastructure (OpenStack) with regards to assured deletion, culminating in open research problems. Section 5 briefly discusses the insights from the two models, positing an initial architecture for assured deletion in the cloud, while Section 6 concludes the paper.

2. ASSURED DELETION

While data security is commonly related to protecting and preserving data privacy, it is important to point out that data disposal is also an essential aspect of managing and protecting sensitive data. It is a key element in assuring confidentiality. Incomplete data deletion may lead to inadvertent exposure of tenants' sensitive information.

Assured deletion methods are different and behave differently. According to [12, 41, 39], assured deletion is achieved when an adversary with access to the system is no longer able to recover deleted data from the system, while [40, 38, 43] also suggest that data is assuredly deleted when it is permanently inaccessible to anyone upon request after it has been deleted. Throughout our discussion, we assume data is assuredly deleted when it can no longer be accessed or recovered or constructed to have any useful meaning after it has been deemed deleted.

Adversarial Models

In the following subsections, we present and describe our two adversarial models. In the first model, we consider a scenario where a tenant uses the services of a dishonest cloud provider while the second model considers a scenario where the cloud provider is honest. In both scenarios, we assume that cloud tenants desire to have their data assuredly deleted from the cloud infrastructure. The difference between the two models is that, in the first model the cloud provider is acting as an adversary while in the second model the provider is not an adversary and is willing to provide assured deletion as a service. Other attacks (e.g., side channel attacks) are out of our scope since there is already a substantial amount of research on such attacks [23, 26, 45].

Dishonest cloud provider

In this scenario, we consider a situation where a cloud tenant is using a public cloud provider for storage and data processing services. The tenant outsources most of his/her data to the cloud but is suspicious and sceptical about the provider's data disposal responsibilities. We assume that the tenant is aware of the risks involved in incomplete or impartial deletion of data and does not want to be a victim of data leakage. Without losing any benefits of using the cloud or incurring any extra cost, the tenant wants to ensure that data will always remain safe even after deletion. We also assume that the provider is not only curious about tenants' data but has some other malicious tenants who are also interested in getting hold of other tenants' data. During provisioning of services the provider's insiders (e.g., system admin) may probe freed resources or decommissioned servers for tenants' data. A malicious tenant may request more resources (e.g., processing server) from the cloud provider, but before writing any data to the availed resource, the malicious tenant probes the provided resource for any sensitive data that may have been left behind by a previous user.

Honest cloud provider

For our second model, we consider a scenario where a public cloud provider is honest but prone to accidental data leaks due to incomplete deletion. Due to escalating number of incidents of data leakages in the cloud, we assume that our provider is conscious about reputation and does not have any intentions to leak tenants' data. We also assume that the provider has other security mechanisms in place to protect the tenants' data from other attacks which could lead to data loss. Additionally, it is in the interest of the provider to provide confidentiality and comply with legal and standard regulations. In spite of the cloud provider's good intentions, malicious tenants may randomly probe their resources for partially deleted data. Again, unless data is completely deleted from the infrastructure, malicious attackers may target decommissioned machines from the cloud provider to steal data.

In the next two sections, we systematise the aspects of assured deletion in the cloud using the two scenarios described above. We review each scenario under the following aspects: requirements for assured deletion, existing approaches and challenges of assured deletion in the cloud, and afterwards compare insights from the two analyses.

3. DISHONEST CLOUD PROVIDER

In situations where the tenant is the one interested in assured deletion and the provider is dishonest, the tenant has to carry the burden of ensuring that data is inaccessible after deletion. From the dishonest cloud provider model above, we draw and summarise the tenants' requirements for assured deletion in the cloud. These are requirements to be considered when a cloud user does not trust the provider.

3.1 Requirements for Assured Deletion

Fine-grained: Deletion should be fine-grained in that only the target data is deleted while other data remains safe and accessible to the tenant. Fine-grained deletion gives the user more control over what to delete in the cloud, therefore, reducing cost of deletion operations.

Usability: User's daily work and use of the cloud should not be affected. Assured deletion should be easy to attain without accumulating any usability cost or affecting the user's productivity.

Cloud Computation: Use of data should not be affected, that is, cloud tenant should continue to work with data as before without any problems. They should be able to complete necessary data operation, for example, sorting and searching.

Complete deletion: Assured deletion requires that the deletion should affect all copies of data associated with the deleted data including the metadata.

Timeliness: Deletion should be completed promptly; deleted data should not be accessible from the environment immediately after deletion is complete.

From a tenant's perspective is it always challenging to assure deletion in the cloud since there is little that a tenant can do – tenants do not have access to the actual physical infrastructure, therefore, they have limited options. In most cases, tenants are forced to explore other complete deletion approaches that do not require modification of the infrastructure. These solutions are mostly based on cryptographic schemes. In the next section we discuss encryption as a solution to guarantee deletion in the cloud. First, we survey and discuss existing encryption approaches to assured deletion in the cloud and then present a discussion of the challenges and limitations of those solutions against the requirements mentioned above. Later, we present the areas that could be explored in order to improve such solutions.

3.2 Existing approaches

In this section, we present existing encryption solutions proposed to guarantee deletion in the cloud. Our aim is to compare existing approaches through a discussion of the assured deletion requirements we outlined above.

As mentioned earlier, deletion is assured when deleted data is irrecoverable or not accessible. All prior work in this area focuses on the scenario where a cloud provider is only offering storage services. To assure deletion using encryption, sensitive data is encrypted before being outsourced to the cloud. Encryption keys are kept secret from the cloud provider. When data reaches the end of its lifetime, before it is deleted, encryption keys are securely destroyed, then data is deleted from the cloud [22, 42]. Consequently, without encryption keys data is deemed inaccessible.

One approach which uses cryptography to assure deletion is FADE [49]. It supports policy-based assured deletion, whereby each file has its own access policy which is also associated with a control key. Before being farmed out to the cloud, files are encrypted with data keys and then control keys. Encryption keys are kept secret and managed independently by a key escrow system. Assured deletion is achieved when the associated file's access policies are revoked. In this approach, fine-grained deletion is achieved since it allows tenants to only revoke polices of the target file or data that needs to be deleted. Assured deletion of a file is also immediately achieved when the access policy is revoked. However, this approach does not consider a scenario where multiple versions of a file exist, which is a case in most cloud setups.

FadeVersion [40] aims to extend FADE by considering a case where multiple versions of a file exist. Each file version has its own encryption key which, when revoked, the version is assumed to be assuredly deleted. Like FADE, fine-grained assured deletion can be achieved by only revoking the access policy of the version one wants to delete. Again, assured deletion is achieved as soon as access policies are revoked. Nevertheless, it should be noted that this approach increases the number of encryption keys for the third party to manage.

Mo et al. explore the feasibility of assuredly deleting data using encryption without involving a third-party [27]. Encryption keys together with sensitive data are outsourced to the cloud provider but are inaccessible to the provider. The authors claim to prevent any potential data privacy breach by using a multi-layered key structure, called Recursively Encrypted Red-black Key tree (RERK). RERK assumes the responsibility of securing data and encryption keys stored in the cloud. The cloud user still maintains a master key or metadata which helps to manipulate encryption keys and data stored by the provider. Fine-grained deletion is supported but the issue of deleting multiple copies remains unaddressed.

3.3 Challenges and limitations

While these solutions do satisfy some of the requirements for assured deletion, they also have various limitations. We discuss these next.

Key Management

One of the reasons why enterprises and other cloud users have struggled to adopt encryption technology is the overhead performance issues and the issue of key management. When using encryption, the client does not only need to deal with protecting the data but also needs to protect the keys themselves. This has required users to adopt third-party key management systems. FADE and FADEVersion base their trust in a key escrow system to manage keys. However, if one cannot fully trust the cloud provider, should this not place the same doubt on these third-party services? These services are vulnerable to the same adversaries as the cloud providers. Moreover, in practice they may introduce some bottlenecks hence reducing performance. For instance, when using a third party provider, one first has to securely access the key before one can access the data. Sometimes this process can delay and even affect performance of the service.

Another challenge is deciding granularity of the keys, whether one key should be used for all or whether each file should have its own key. Mo et al.[27] and Rahumed et al.[40] consider fine-grained deletion in their solutions, both solutions suggesting that each file should have its own encryption key. Nonetheless, this increases the number of keys and may be cumbersome to manage hence leading to usability problems.

Table 1: Summary: Open research areas

Emerging Solution (EM)	Challenges
EM1- Homomorphic Encryption	Impractical for cloud applications.
	Causes a lot of computational overheads.
EM2- Partial Homomorphic Encryption	Support limited amount of datasets[6].
	Does not support all queries.
	Causes a lot of computational overheads.
EM3- Searchable encryption	Does not support all queries.
	Causes a lot of computational overheads [51].
EM4- Trusted Computing	Expensive
	User needs to transfer secret key to the trusted hardware [6].
	Compatibility with current infrastructures.
	Verification and attestation [46]

Data sharing also becomes a problem especially when more than one user needs to access the encrypted data. This is explained later in detail under data sharing subsection. Again, key management systems are sometimes proprietary and support a limited number of cloud providers, cloud tenants may struggle finding an appropriate key management system.

Usability

Integration of encryption into systems has always shown to have impact on users [53, 48, 21]. Research has shown that users struggle to use encryption systems and are likely to make mistakes especially with encryption keys. As a result, this may lead to cloud users making mistakes when deleting keys as part of guaranteed deletion. Users are bound to make mistakes, for example, a user deleting the wrong keys or forgetting to encrypt a file after use. Another concern is the issue of effort, i.e., the addition of extra steps into the process of completing a daily task (often such extra steps are circumvented by users when they get in the way of daily activities [1]. Again, users are not only required to understand why such solution is in place but also, their responsibilities to make it work effectively.

Limited Data Usage

When data is encrypted outside the cloud, there is little to do with it unless keys are also shared with the cloud provider or if the cloud is only used for storage (e.g., Storage as a Service). Encryption limits data use. Current solutions do not mention how this issue will be addressed if the cloud tenant also uses the cloud to process data. Current cloud applications do not yet accommodate the use of encrypted data. When encrypted, data loses some of its properties such as the ability to be searched or sorted since it is in unrecognised cipher text. A naive solution would be to download it first before use. Still, this can be burdensome on the client hence reducing productivity.

Although there have been some proposed solutions [28, 50, 13] to tackle the above-mentioned issues, most of them are still in their infancy and are not yet practical to be adopted in cloud applications. For example, most of the homomorphic encryption schemes that have been proposed only support a limited number of operations [18].

Data Sharing

Encryption makes sharing data difficult especially when all involved parties need to access the same data. Current cryptographic schemes depend on using a single set of secret keys, therefore, allowing access to a single user. In order to share data, users end up sharing such keys which can be costly during key revocations as it requires all the data to be re-encrypted [17]. Moreover, this leaves data and keys vulnerable.

One of the benefits of cloud computing is high availability, the ability to access data regardless of location and time. However, with encryption in place this can become a challenge as users may need to share encryption key between devices hence putting keys at risk.

3.4 Future Directions

Since encryption is a de facto solution for scenarios where the tenant does not have full trust in the provider, research work could focus more on trying to improve the current encryption schemes (e.g., searchable and homomorphic encryption schemes) that compute data. Development of encryption ready cloud applications could be an area to be explored.

Trusted computing is an alternative area to be explored or to be used alongside encryption. Cloud providers could offer secured hardware containers within their cloud infrastructures[5].

In Table 1 we summarise some emerging solutions with their respective limitations.

4. HONEST CLOUD PROVIDER

As we mentioned earlier, there are many reasons why a cloud provider may want to provide assured deletion as a service; it might be to comply with standards or to enhance reputation in the market. In this section, we summarise the provider's requirements for assured deletion using the honest cloud provider model.

4.1 Requirements for Assured Deletion

Service availability: Assured deletion should not negatively affect services offered to other tenants or even to the tenant who requested the deletion. Any service disruption would cost the provider.

Complete deletion (irrecoverable and inaccessible): Data should be removed from all layers and components that handle data. Data residing in temporary locations such as buffers and RAM should also be wiped out completely. Deleted data should not be accessible after deletion, either to the provider or tenants. Data should be completely wiped from the provider's infrastructure.

Deletion of all backup copies: The provider should be

able to completely delete all backup data that is no longer needed or required by the tenants. This includes data stored locally and remotely.

Fine-grained deletion: Provider should be able to assure deletion on target data (data which is to be deleted and requested by the tenant). When a tenant requests the provider for data deletion, the provider should only be able to completely delete requested data without affecting the tenant's other data. Fine-grained deletion reduces cost of secure deletion operations.

Delete latency: Timeliness to guarantee deletion - this refers to the time agreed by the provider to assure deletion to a cloud tenant or the time required by any regulatory standards to assure deletion. The provider should be able to completely delete data within this time.

Error handling: Deletion procedure should complete without any errors and if errors do occur, the deletion procedure should be able to recover and complete within a reasonable time.

Proof of deletion: An honest cloud provider should be able to prove deletion to tenants, attesting that data deletion has completed successfully. For example, if a deletion procedure has completed successfully, it should return a signature which can be verified by the client.

In the absence of existing literature targeting this scenario, we analyse OpenStack, an existing cloud infrastructure. We opted for OpenStack as an example because it is the most used open source cloud platform. Different cloud features and characteristics pose different challenges to assured deletion. We discuss these challenges next against the requirements mentioned above.

4.2 Case Study: OpenStack

OpenStack is a fully distributed open source infrastructure as a service (IaaS) cloud platform used for building public and private clouds infrastructures. NASA and Rackspace were the first companies to develop and contribute to the OpenStack project but now it has more than 30 contributing companies [31].The main core of OpenStack is fully developed in Python and has powerful APIs which allow it to be interfaced with other cloud computing platforms such as Amazon EC/S3, making it possible to offer hybrid cloud services. OpenStack is made up of several components or projects, which include:

OpenStack Compute (Nova): The main component of Openstack; its main purpose is to provide virtual instances on-demand.

Openstack Image Service (Glance): Glance is a virtual image repository service which provides services for discovering, registering and retrieving virtual images in OpenStack.

Network Service (Neutron): Provides network connectivity between interface devices in OpenStack.

OpenStack Object Storage(Swift): A highly available and distributed Object Storage.

Block Storage (Cinder): Provides persistent block storage for virtual instances.

Other key components of OpenStack include *Keystone* and *Horizon*. Authentication and authorization for OpenStack are handled by the *Keystone* component while *Horizon* is a web based graphical user interface for OpenStack components [32]. Fig 1 shows the key components of OpenStack.

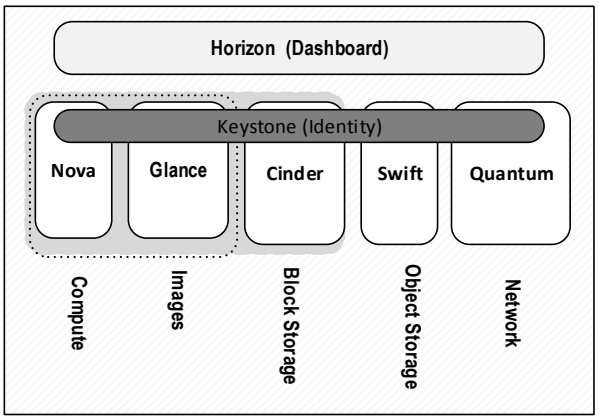


Figure 1: OpenStack Core Components

4.2.1 Assured Deletion Challenges

Like other cloud platforms, OpenStack has features and components that lead to challenges in terms of assured deletion. We discuss these challenges next.

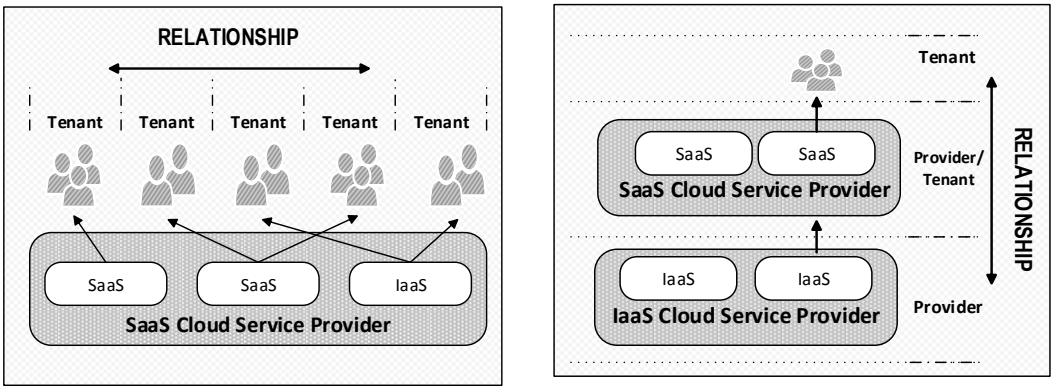
Many Components

OpenStack has a very complex architecture with many different components. Numerous services and components capture, maintain or reference tenants' sensitive data [33]. Besides OpenStack storage components, tenants' sensitive data may also be found in these components: *Neutron*, *Glance*, *Keystone* and *Nova*. These data include information about the instances, storage volume data and public keys while sensitive metadata may include organization names, internally generated private keys and users' real names. Although OpenStack is expected to delete this data when required to do so, some data may still remain in the platform after deletion[33]. Layers, virtual and physical components may trap data and making it difficult to locate during a deletion request. As a result, it is difficult to completely delete data from OpenStack due to the complexity of the architecture.

Virtualization Technology

In OpenStack, deletion is also subject to the virtualization technology used. To explain this challenge, we discuss a scenario in which a tenant launches and terminates a virtual instance. When a tenant launches an instance, OpenStack makes a copy of the base image from *Glance* to the local disk of the compute node. A new ephemeral volume is then created and attached to the running instance. For persistent writing, block storage can then be attached to the instance. When an instance is decommissioned, the metadata of the original image is updated with the new metadata from the running instance. All unsaved data is then saved to the block storage before it is detached. The previously created ephemeral storage is then deleted before memory and CPU resources are released[34].

One of the requirements for assured deletion for the provider is the ability to make data inaccessible within an agreed time. However, deletion in OpenStack may not always be



(a) Horizontal Multi-tenancy relationship (b) Vertical Multi-tenancy relationship

Figure 2: Multi-tenancy relationship in the cloud

complete. The destruction of images and the ephemeral storage is dependent on the hypervisor and the compute plug-in used [19, 33]. Some plugins (e.g., libvirt) would not overwrite the memory block that was previously used by the running instance. It is assumed that the dirty memory extents will not be made available to other tenants, but this cannot be guaranteed. It is difficult for the provider to predict or estimate when the dirty memory would be overwritten. Also, in practice hundreds of instances are launched and decommissioned all the time. Therefore, it is not always feasible for the provider to confirm or provide proof of deletion after every terminated instance.

Live Migration

There are two types of migration supported by OpenStack: True Live migration and Non-live migration (or simple migration). Unlike other cloud settings where live migration is a load management feature, OpenStack has this as an operational management feature. For example, migration may be required when performing upgrades that need system reboot of the physical server. Live migrations in OpenStack take place under controlled conditions and cloud administrators initiate them. In spite of this, migration still increases data surface area. For example, after the virtual instance is successfully copied to a new location, there might still be data left on the previous host. OpenStack currently does not have any feature to check if all data from the source host has been deleted. Although the hypervisor might delete data after migration, the provider cannot assure that data and metadata of the migrated virtual machine is completely removed from the source environment [33].

Virtualized Storage

Deletion of data from *Swift* is a complicated process, which may not result in complete deletion. Like other virtualized storages, *Swift* does not provide direct access to raw blocks of data but rather provides blobs of data(data objects) which represents "volumes" and they are accessible through an API [2].Data is stored as objects and can be stored anywhere in the storage or even replicated to other *Swift* clusters. Since this is virtualized, the pool of storage provided may span several physical servers or drives. This allows data to be scattered all over the storage infrastructure. To access data, OpenStack makes multiple requests to these storage servers through a *Swift* API. It is difficult to assure deletion because the physical location of data is abstract and may not be known, and also, deletion methods such as overwriting will not overwrite an object but rather create one. Virtualization layers may also possess some data that may be left behind since deletion is not done at all layers. Deletion of backup copies in other *Swift* clusters may not happen in time or at all.

Horizontal Multi-tenancy

In the cloud tenants share the same physical infrastructure and each tenant has his/her own relationship with the provider. Multi-tenancy relationships in the cloud can either be horizontal or vertical depending on the services provided. Horizontal multi-tenancy relationship exists when a provider is offering their services directly to multiple tenants simultaneously as shown by Fig 2a while vertical multi-tenancy exists when a provider is offering services to other multiple tenants while it is also leasing computer resources from another provider as shown by Fig 2b. In these kind of setups, it can be challenging for the provider to assure deletion since tenants' data may be residing in the same location or even tangled together. In horizontal multi-tenancy, it is possible that when a tenant is decommissioned, their storage partition will not be securely wiped out completely before it is made available to new tenants. This may be due to a number of reasons: (1) the hypervisor used as explained earlier in the section under virtualization technology, (2) secure deletion techniques like overwriting may not be feasible between provisioning of services, (3) the secure deletion methods used may not be fine-grained and leave other tenants' data at risk during the deletion.

Vertical Multi-Tenancy and Third Party Backup Providers

OpenStack interfaces allow it to be integrated with other cloud computing platforms either for storage or computing services. For example, OpenStack can be interfaced with Amazon S3 for storage. This would form a vertical multi-tenancy relationship. It becomes very difficult to delete data that leaves Openstack because it may be out of reach. An-

other challenge is that if these APIs fail then the provider has no other way to verify the deletion in the other infrastructure. Complete deletion may require the involvement of the third party, which may add extra cost to the provider; again, it may not happen as the other provider may face the same problems highlighted in this paper. Deletion latency could also be a problem.

Delayed Deletion

Another cloud feature that challenges the requirements for assured deletion is the delayed deletion feature. By default, most providers use a garbage collection method of deletion whereby deleted data is only marked for deletion but not removed from the system. This feature is enabled to protect tenants' data from accidental deletion operations or administrative errors. When a tenant deletes a file, the cloud system will only mark it for deletion and remove it from the tenant's interface while it remains in the cloud. Sometimes, providers have to delay deletion because of legal obligations or policies. Therefore, deletion does not happen as soon as it is requested.

In OpenStack when delayed deletion is enabled, deleted accounts, images, volumes and other data are not removed from the platform upon request [2, 19]. By default, there is a 7-day delay before an image is removed from the platform. When an image is deleted, it is marked for deletion and given an expiry date. Although it is removed from the tenant's reach, it is still part of tenant's data. The removal of the image is then left for the reaper process - a periodical background process that is responsible for removing all data marked for deletion after retention time has expired [35]. It is difficult to prove this deletion or estimate the time it takes to completely delete data from the provider. Data may still be accessible after deletion.

Execution Errors

OpenStack is not immune to failures. Its activities are vulnerable to errors. Migration, creation of instances, volume attachments, deletion and many other processes may fail one way or another. Failures may be due to network errors, hypervisor errors, inadequate resources or even administrative errors. When an instance fails during provisioning, the service provider cannot guarantee that the tenant's data that was being processed will be destroyed completely. The same problem exists with components: when a component fails, the provider cannot guarantee whether data was completely removed from the component before it encountered errors. In this case, deletion assurances may be difficult to guarantee.

High Availability

According to OpenStack documentation [30], it can meet high availability requirements for its own infrastructure services by up to 99.99% uptime. Images, containers, objects, volumes and metadata can be replicated in OpenStack to allow high availability requirements. A process called the replicator is responsible for producing multiple copies of data and syncing all data. Through Cinder Backup API, OpenStack allows volume replication, multiple backend backup system with tiering technology and compression. This API also allows the import and the export of tenants' backup service metadata. It may be difficult for the provider to guarantee that all data copies and metadata will completely be deleted when tenants request for deletion.

Challenges that depend on deployment

There are other challenges that manifest on OpenStack but are dependent on deployment. These challenges include: Underlying hardware (e.g., the use of Solid State Drives (SSDs)), Storage Tiering and Thin Provisioning, Multiple physical locations, Offline Backup, Different storage media and Third-party providers.

4.2.2 Summary

Different cloud features pose different challenges with regards to assured deletion. We have outlined and discussed these challenges against the requirements mentioned in the previous section. In OpenStack, these challenges include: Live migrations, virtualized storage, execution errors, vertical and horizontal multi-tenancy, delayed deletion, high availability and virtual technology. Table 2 presents a summary of the list of cloud features and the key challenges they pose with regards to assured deletion. The second column shows the requirements impacted by these challenges.

In the next section, we present some future research directions for assured deletion in the cloud with regards to the provider wishing to provide assure deletion as a service.

4.3 Future Directions

This section is motivated by the requirements and challenges mentioned in the previous sections. We have classified these possible areas of research according to the requirements we mentioned earlier. Our suggestions remain focused on the public cloud paradigm.

Service availability

Service availability is very important to cloud providers and tenants. Interruption of service for seconds could mean huge financial losses for the provider. Research could be done to find ways of assuredly deleting data without affecting the service. For example, applying secure deletion methods (e.g., overwriting) without interrupting the service or deleting other tenants' data. Research should look into solutions which do not only rely on secure deletion methods (e.g., overwriting) which depend heavily on the properties of the underlying physical storage since physical control over infrastructure is no longer feasible [11]. Again, in order to assure deletion without any service disruption, data isolation mechanisms could be developed to isolate data that would require assured deletion. Restrictions could be applied to sensitive data. For example, sensitive data could be restricted to certain places in the cloud.

CloudFence [37] and Cloudfilter [36] are some of the approaches that have been proposed to restrict data movement in the cloud. Cloudfilter proposes a practical service-independent system that uses policies to restrict data movement between enterprise and cloud service provider while Cloudfence focuses on confining sensitive data to a single defined domain thus preventing the propagation of marked data to other parts of the cloud infrastructure. Researchers could leverage these approaches and use the tracking capabilities to confine sensitive data to locations where assured deletion methods could be applied in the cloud. For instance, sensitive data could be restricted to disks that allow scrubbing.

Table 2: OpenStack cloud features and their respective challenges against assured deletion requirements

OpenStack feature	Requirement Challenged	Key Challenge
Infrastructure	Complete deletion Deletion of all backup copies Deletion latency Proof of deletion	Multiple Components
Virtualization	Complete deletion Deletion of all backup copies Fine-grained deletion Error Handling	Virtualization Technology Virtualized Storage Underlying Hardware Virtual Instance Operations Multiple Dynamic Logical layers
Multi-tenancy	Service Availability Complete deletion Fine-grained deletion	Horizontal Multi-tenancy Vertical Multi-tenancy
On-Demand Elasticity	Complete deletion Deletion of all backup copies Error Handling	Live Migration Storage Tiering and Thin Provisioning Execution Errors
Backup and High Availability	Complete deletion Deletion latency Proof of deletion	Multiple Locations Offline backup Storages Different Storage Media Third-Party Providers Delayed Deletion

Complete deletion (irrecoverable and inaccessible)

Research could focus on coming up with better ways of knowing the location of data that needs to be deleted. Data life cycle should be tracked around the cloud in order to help providers make better decisions on which methods of assured deletion to apply.

Reardon et al.[41], presented a survey which focused on analysing and discussing methods of secure deletion. It is stated that in order to know which secure deletion approach to use, one has to know the properties of the environment before deciding the most suitable method. In terms of the cloud, it would be ideal to track and locate data during its life cycle that is, getting all the relevant information needed for deletion so as to help the provider with a better method of assuring deletion.

Deletion of metadata is another area that could be explored. To completely delete data also means getting rid of the metadata associated with data that is being deleted. These metadata do sometimes hold sensitive information. Solutions could include finding ways of deleting metadata from all places in the cloud. Diesburg et al.[15] proposed how metadata could be deleted from NAND flash drives and hard drives.

Deletion of data could also be performed at every layer of the cloud to assure that data is completely removed. Secure deletion methods could be refined to work in these layers to guarantee deletion. Again, using provenance, data could be tracked up to the physical layer in order to capture the mapping between virtual and physical resources. This would also give cloud users some sense of transparency and show them how virtual locations and physical static server locations are linked or mapped together. Also, this mapping will allow deletion done on the virtual environment to be confirmed at the physical level hence enabling proof of deletion.

In a complex system like the cloud, researchers could develop security policies for each component in the cloud rather than having policies tied only to data. As mentioned earlier, data in the cloud may get trapped in different cloud components and never be assuredly deleted. New solutions can focus on developing a set of policies that could be implemented by each individual component to make deletion easy and more secure.

Deletion of all backup copies

To assure availability and durability in the cloud, data is replicated in the cloud several times. During deletion, it can be a challenge to guarantee that all copies are deleted. Also, data can get trapped in different cloud layers and components and be left behind during deletion. Research could focus on tracking this data and keeping record of all the places it could be so that during deletion, such data could also be removed. Proof of Retrievability (PoR) and Provable Data Possession (PDP) are areas that could be considered to verify the number of copies in the cloud. Traditional PoRs [9, 24] only demonstrate that the provider has the data not how many copies exist in its possession. Benson et al. [8] looked into verifying that data is present in multiple locations (e.g., multiple disks) within the cloud. Their framework complemented Bowers et al. [10] who looked at the same verification but on a single geolocation or datacentre. Both of these approaches could prove to be vital in providing the location of data, that is, location within the datacentre and location of data with respect to geographical locations. They can also be used to verify number of copies before deletion so as to help the provider in deleting all copies and verifying that the same number of copies is deleted after deletion request. Watson et al. [52] also explored the possibilities of verifying that the provider is storing files at the requested location which can also be leveraged to prove deletion.

Another approach that could help to verify deletion of multiple copies of data previously held by a provider is proposed by Barsoum et al. [7], aimed at verifying that the

Table 3: A mapping of assured deletion requirements and research areas that could be explored to satisfy them.

Deletion requirements	**Summary of possible areas to be explored**
Service availability	Tenants Isolation
	Controlled movement of data in the cloud infrastructure
Complete deletion	Use of secure deletion methods which are suitable and appropriate for the cloud.
	Secure deletion of metadata
	Deletion of data at every cloud layer
	Secure deletion policies for each component
Deletion of all backup copies	Keeping track of number of copies
	Verifying the number of copies before and after deletion
	Data provenance during migration
Fine-grained deletion	Data provenance for sensitive data.
	Isolation of data that needs to be securely deleted.
	Use of secure deletion methods which are suitable and appropriate for the cloud.
Deletion latency	Exploration of secure deletion mechanisms that can delete data promptly.
Error handling	Resilient mechanisms for secure deletion in the cloud.
Proof of Deletion	Verification and attestation of deletion in the cloud.

provider has more than one copy of outsourced data in its possession. They extended existing PDP [3] that focused on a single copy of data. Under assured deletion, this work can also be used to verify whether the provider still possesses other copies after deletion.

To ensure deletion of data left behind after migration, research could focus on capturing all the necessary information involved during migration. This may include data that moved and the data that failed to move during migration. This could also give the provider an opportunity to delete all data in case of errors and actually confirm it.

Fine-grained deletion

Multi-tenancy is an important feature of the cloud – multiple users can share resources (e.g., storage, tables) hence reducing cost for the provider. This sharing of resources can lead to data being tangled and mixed up hence proving to be a challenge during deletion. As stated before, applying secure deletion mechanisms may lead to service disruption for other tenants using the cloud in case the secure deletion method deletes other tenants' data. Again, assured deletion at a fine-grained level is needed. For example, the provider should be able to completely delete tenants' data that need to be deleted while other data remains untouched. Cachin et al. constructed a deletion scheme that aims to provide fine grained deletion. The scheme maintains a collection files and provides deletion of the files that need to be deleted [11].

Not all data stored in the cloud is sensitive therefore data provenance could be explored in depth to track sensitive data and only allow data with the same level of deletion requirements or retention duration to be stored in same locations. These might be media which allow secure deletion mechanisms.

Another area to explore is that of assured deletion methods (e.g., overwriting). Research should look into how these methods could be applied in the cloud, for example, how an overwriting method could be applied to few bits of storage rather than the entire disk partition.

Deletion latency

Time to guarantee deletion can depend on many things, for instance, the distance between data centres or the number of copies that need to be deleted. Research could focus on these properties to develop and design better mechanisms which could assure deletion within a reasonable time. Assured deletion procedures could be timed and the elapsed time could be used as part of evidence in proving deletion. Research could also focus on finding out how secure deletion methods could be improved to complete promptly in cloud settings.

Error handling

Research could look into improving the current secure deletion methods which can recover during failures and still completely delete data. For instance, Diesburg et al. [14] developed an assured deletion framework for the storage data path that could handle crashes. Such works could be extended to the cloud and incorporated with the cloud system to report any deletion crashes. Deletion mechanisms in the cloud could be improved to report failures.

Proof of deletion

Methods are needed to attest whether deletion mechanisms complete deletion successfully without errors. These could provide building blocks to proving deletion. Deletion attestations could either be done at software level or hardware level depending on deletion requirements. Backes et al.[4] proposed a verification mechanisms which a cloud user could use to force a cloud provider to attest deletion of data.

In Table 3 we summarise the above areas of research and the requirements they may satisfy. Column 1 presents the requirements while column 2 outlines the areas of research.

5. DISCUSSION

In Fig 3 we present a conceptual architecture for assured deletion which targets both scenarios. We assume a model that is protected against a peek-a-boo adversary as described by Reardon et al. [41]. The conceptual architecture consists of the following main components:

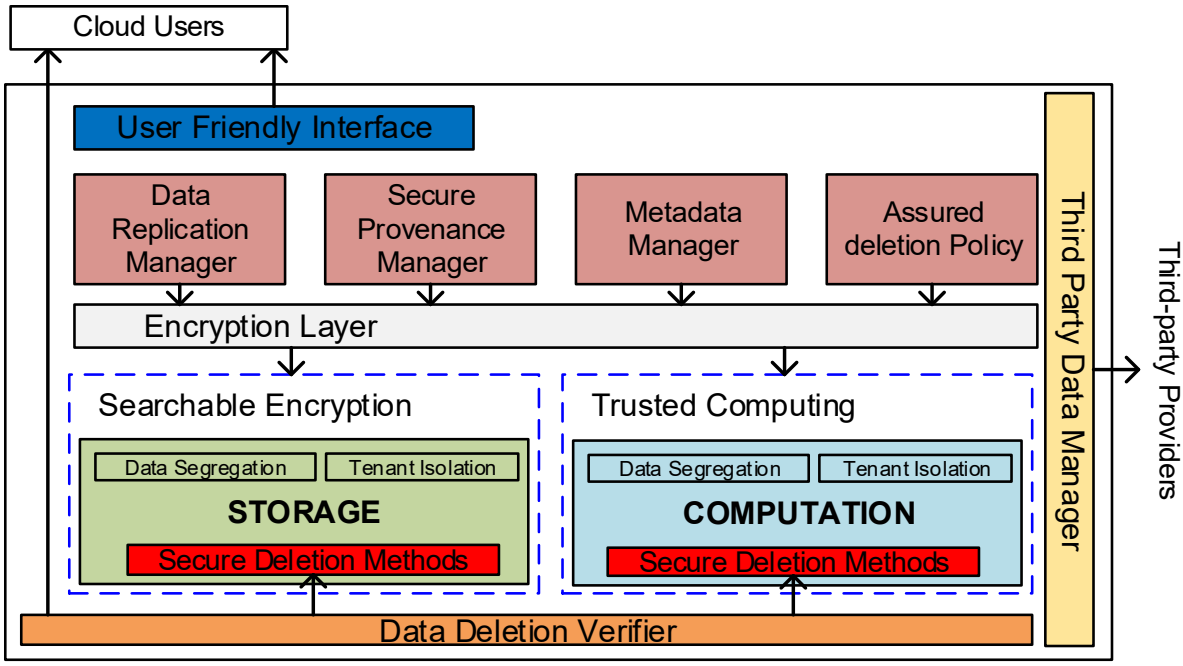


Figure 3: Assured Deletion Conceptual Architecture

- **Data replication manager**
  It is responsible for managing data replication, keeping track of how sensitive data is replicated in the cloud system. This ensures that sensitive data is only replicated according to users' requirements or data policies.
- **Secure provenance manager**
  It is responsible for the tracking of all sensitive data in the cloud throughout its life cycle. This ensures that location of all sensitive data is tracked throughout the cloud system enabling the provider to locate and delete data when required. This will make sure that data trapped between the cloud layers is also deleted.
- **Metadata manager**
  This component is responsible for the management of information about data, together with the secure provenance manager, it ensures that all metadata of deleted data is also removed from the system.
- **Assured deletion policy module**
  This module makes sure that all data deletion policies are enforced in the cloud system. For example, if sensitive data is to be restrained to a specific location, this module will enforce that particular policy.
- **Third-party data manager**
  This component is responsible for the management of third party activities in the cloud, ensuring that deletion policies (enforced by the policy module) are respected by third party services. For example, making sure that data marked sensitive does not leave the cloud system where secure deletion can be guaranteed.
- **Deletion verifier**
  This is a trusted component of the cloud which is responsible for verifying deletion whether from the storage or the computation area. When a tenant poses a challenge to an honest cloud provider, this module will produce a signature which can be verified by the user to prove deletion.
- **Encryption layer**
  This layer handles all encryption related procedures including data operations such as searching encrypted data.

In scenarios where the provider is dishonest, this architecture should provide tenants with the ability to completely delete data effortlessly without encountering any difficulties. Assured deletion interfaces could be incorporated and be made easy to use in the cloud.

For honest providers, data should be segregated at all levels of the cloud to avoid service disruptions during complete deletion, for instance, tenants' data should be isolated to allow secure deletion without deleting other tenants' data. By using trusted computing modules, tenants should be allowed to process their sensitive data, and verify deletion within the cloud. Data should be tracked and monitored throughout the cloud environment to allow deletion of all copies. The new architecture should allow secure deletion of metadata when associated copies are deleted, and also allow secure deletion methods to be applied in all layers on the cloud. This architecture may change as homomorphic encryptions develop and become more achievable.

6. CONCLUSION

Although assured deletion is a significant hurdle for adoption of public clouds, it could also become a differentiator in the market. Allowing cloud users to control and verify

how their data is handled is essential for even greater adoption. We have shown the importance of assuring deletion in the cloud and presented assured deletion for both the cloud tenant and the provider. In cases where a dishonest cloud provider is used, we have surveyed and discussed existing solutions against requirements and outlined their limitations. For the honest provider, we have outlined assured deletion requirements for the provider, reviewed current infrastructures then provided a systematization of assured deletion challenges their features pose with regards to assured deletion. The open research directions discussed in this paper are a stepping stone in tackling the challenge of assured deletion in the cloud, and provide a research agenda for both our own research and that of the wider community.

7. ACKNOWLEDGMENTS

We want to thank Alessandro Sorniotti for his invaluable time and effort for improving this work. We would also like to thank all the anonymous reviewers for their helpful comments and suggestions.

8. REFERENCES

[1] A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[2] J. Arnold. *OpenStack Swift: Using, Administering, and Developing for Swift Object Storage*. O'Reilly Media, 2014.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609. Acm, 2007.

[4] M. Backes and F. Bendun. Poster: Forcing the cloud to forget by attesting data deletion.

[5] M. Barhamgi, A. K. Bandara, Y. Yu, K. Belhajjame, and B. Nuseibeh. On Protecting Privacy in the Cloud. *IEEE Computer*, 2016.

[6] M. Barhamgi, A. K. Bandara, Y. Yu, K. Belhajjame, and B. Nuseibeh. Protecting privacy in the cloud: Current practices, future directions. *Computer*, 49(2):68–72, 2016.

[7] A. F. Barsoum and M. A. Hasan. Provable possession and replication of data over cloud servers. *Centre For Applied Cryptographic Research (CACR), University of Waterloo, Report*, 32:2010, 2010.

[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*, pages 73–82. ACM, 2011.

[9] K. D. Bowers, A. Juels, and A. Oprea. Proofs of retrievability: Theory and implementation. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 43–54. ACM, 2009.

[10] K. D. Bowers, M. Van Dijk, A. Juels, A. Oprea, and R. L. 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*, pages 501–514. ACM, 2011.

[11] C. Cachin, K. Haralambiev, H.-C. Hsiao, and A. Sorniotti. Policy-based secure deletion. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 259–270. ACM, 2013.

[12] J. Chow, B. Pfaff, T. Garfinkel, and M. Rosenblum. Shredding your garbage: Reducing data lifetime through secure deallocation. In *USENIX Security*, pages 22–22, 2005.

[13] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 79–88. ACM, 2006.

[14] S. Diesburg, C. Meyers, M. Stanovich, M. Mitchell, J. Marshall, J. Gould, A.-I. A. Wang, and G. Kuenning. Trueerase: Per-file secure deletion for the storage data path. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 439–448. ACM, 2012.

[15] S. Diesburg, C. Meyers, M. Stanovich, A.-I. A. Wang, and G. Kuenning. Trueerase: Leveraging an auxiliary data path for per-file secure deletion. *ACM Transactions on Storage (TOS)*, 12(4):18, 2016.

[16] M. V. Dijk, A. Juels, and A. Oprea. Hourglass schemes: how to prove that cloud files are encrypted. *Proceedings of the ...*, pages 265–280, 2012.

[17] C. Dong, G. Russello, and N. Dulay. Shared and searchable encrypted data for untrusted servers. *Journal of Computer Security*, 19(3):367–397, 2011.

[18] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[19] T. Fifield, D. Fleming, A. Gentle, L. Hochstein, J. Proulx, E. Toews, and J. Topjian. *OpenStack Operations Guide*. O'Reilly Media, 2014.

[20] A. A. Friedman and D. M. West. *Privacy and security in cloud computing*. Center for Technology Innovation at Brookings, 2010.

[21] S. Furnell. Why users cannot use security. *Computers & Security*, 24(4):274–279, 2005.

[22] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy. Vanish: Increasing data privacy with self-destructing data. In *USENIX Security Symposium*, pages 299–316, 2009.

[23] G. Irazoqui, T. Eisenbarth, and B. Sunar. Sa: A shared cache attack that works across cores and defies vm sandboxing—and its application to aes. *IEEE: Security & Privacy*, 2015.

[24] 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*, pages 584–597. Acm, 2007.

[25] Ł. Krzywiecki and M. Kutyłowski. Proof of possession for cloud storage via lagrangian interpolation techniques. In *Network and System Security*, pages 305–319. Springer, 2012.

[26] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee. Last-level cache side-channel attacks are practical. In *36th IEEE Symposium on Security and Privacy (S&P 2015)*, 2015.

[27] Z. Mo, Q. Xiao, Y. Zhou, and S. Chen. On deletion of outsourced data in cloud computing. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 344–351. IEEE, 2014.

[28] M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.

[29] D. Ocean. *Data leakage*, 2013 (accessed May 14, 2015). https://www.digitalocean.com/company/blog/resolved-lvm-data-issue/.

[30] OpenStack. *OpenStack High Availability*, 2015 (accessed June 2, 2015). http://docs.openstack.org/high-availability-guide/content/ch-intro.html.

[31] OpenStack. *OpenStack Official*, 2015 (accessed June 2, 2015). http://www.openstack.org/.

[32] OpenStack. *OpenStack Architecture*, 2015 (accessed May 15, 2015). http://docs.openstack.org/openstack-ops/content/architecture.html.

[33] OpenStack. *OpenStack: Data privacy concerns*, 2015 (accessed May 15, 2015). http://docs.openstack.org/security-guide/content/data-privacy-concerns.html.

[34] OpenStack. *OpenStack Image and Instances*, 2015 (accessed May 2, 2015). http://docs.openstack.org/admin-guide-cloud/content/section_compute-images-and-instances.html.

[35] OpenStack. *OpenStack Reaper*, 2015 (accessed May 28, 2015). http://docs.openstack.org/developer/swift/overview_reaper.html.

[36] I. Papagiannis and P. Pietzuch. Cloudfilter: practical control of sensitive data propagation to the cloud. In *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, pages 97–102. ACM, 2012.

[37] V. Pappas, V. P. Kemerlis, A. Zavou, M. Polychronakis, and A. D. Keromytis. Cloudfence: Data flow tracking as a cloud service. In *Research in Attacks, Intrusions, and Defenses*, pages 411–431. Springer, 2013.

[38] R. Perlman. File system design with assured delete. In *Third IEEE International Security in Storage Workshop (SISW'05)*, pages 6–pp. IEEE, 2005.

[39] C. Priebe, D. Muthukumaran, D. O'Keeffe, D. Eyers, B. Shand, R. Kapitza, and P. Pietzuch. Cloudsafetynet: Detecting data leakage between cloud tenants. In *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*, pages 117–128. ACM, 2014.

[40] A. Rahumed, H. C. Chen, Y. Tang, P. P. Lee, and J. C. Lui. A secure cloud backup system with assured deletion and version control. In *Parallel Processing Workshops (ICPPW), 2011 40th International Conference on*, pages 160–167. IEEE, 2011.

[41] J. Reardon, D. Basin, and S. Capkun. Sok: Secure data deletion. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 301–315. IEEE, 2013.

[42] J. Reardon, D. Basin, and S. Capkun. On secure data deletion. *Security & Privacy, IEEE*, 12(3):37–44, 2014.

[43] J. Reardon, S. Capkun, and D. A. Basin. Data node encrypted file system: Efficient secure deletion for flash memory. In *USENIX Security Symposium*, pages 333–348, 2012.

[44] J. Reardon, H. Ritzdorf, D. Basin, and S. Capkun. Secure data deletion from persistent media. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 271–284. ACM, 2013.

[45] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212. ACM, 2009.

[46] N. Santos, K. P. Gummadi, and R. Rodrigues. Towards trusted cloud computing. *HotCloud*, 9:3–3, 2009.

[47] A. Shraer, C. Cachin, A. Cidon, I. Keidar, Y. Michalevsky, and D. Shaket. Venus: Verification for untrusted cloud storage. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pages 19–30. ACM, 2010.

[48] S. W. Smith. Humans in the loop: Human-computer interaction and security. *IEEE Security & privacy*, 1(3):75–79, 2003.

[49] Y. Tang, P. P. Lee, J. C. Lui, and R. Perlman. Fade: Secure overlay cloud storage with file assured deletion. In *Security and Privacy in Communication Networks*, pages 380–397. Springer, 2010.

[50] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.

[51] P. Van Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker. Computationally efficient searchable symmetric encryption. In *Workshop on Secure Data Management*, pages 87–100. Springer, 2010.

[52] 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*, pages 59–70. ACM, 2012.

[53] A. Whitten and J. D. Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Usenix Security*, volume 1999, 1999.