Towards a forensically ready cloud storage service[1]

Theodoros Spyridopoulos and Vasilios Katos
Information Security and Incident Response Unit, Democritus University of Thrace, Greece
{theospyr,vkatos}@ee.duth.gr

Abstract

In this paper we examine the feasibility of developing a forensic acquisition tool in a distributed file system. Using GFS as a vehicle and through representative scenarios we develop forensic acquisition processes and examine both the requirements of the tool and the distributed file system must meet in order to facilitate the acquisition. We conclude that cloud storage has features that could be leveraged to perform acquisition (such as redundancy and replication triggers) but also maintains a complexity, which is higher than traditional storage systems leading to a need for forensic-readiness-by-design.

Keywords

distributed storage forensics, remote acquisition, jurisdiction.

1. Introduction

The imperative need for forensic readiness within an organization can be established not only from the need to incorporate forensic acquisition capabilities within security best practices (Grobler & Louwrens, 2007), but from the fact that "ongoing and pro-active computer investigations are now a mandatory component of the IS enterprise" (Patzakis, 2003). However the marketing momentum surrounding Cloud oriented solutions reminds us the late 1990s e-commerce unprecedented growth, where security amongst other important facets of an Information System was not within the strategic priorities of a newly formed dot com. Within the last 15 years there has been a significant change in culture and information security is nowadays seen as means to help an organization succeed.

Yet it seems that although a bottom-up approach to security is followed during the design phase of a cloud infrastructure, forensic capabilities have not received enough attention. Considering that a user's actions could lead to litigation and as such his/her files – deleted or not – would need to be retrieved from the cloud, there need to be processes in place to support forensic readiness. Forensic acquisition of virtual disks should also be carried out in a way that shared space belonging to legitimate users will not be harvested in order to protect their privacy. This constraint together with

[1] The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013_FP7-REGPOT-2010-1, SP4 Capacities, Coordination and Support Actions) under grant agreement no. 264226 (project title: Space Internetworking Center-SPICE). This paper reflects only the author's views and the Union is not liable for any use the may be made of the information contained therein.

jurisdiction introduces interesting challenges and calls for both technical and non-technical approaches when developing a "cloud-dd" tool.

2. Background

The term cloud refers to an infrastructure that enables convenient, on-demand network access to a shared pool of resources (e.g. storage, networks, servers, applications and services) that can be rapidly provisioned and released (Mell and Grance, 2009). Cloud systems essentially compose a network of distributed clusters forming a pool of resources ready to be used from clients. The physical distance between the location of each cluster varies from some meters in one data center to thousands of kilometers between data centers located in different countries or even different continents. Thus, when someone uses a cloud, her data are distributed in a network of clusters around the world. To achieve distribution of data, cloud systems make use of a distributed file system (Thanh *et al.,* 2008). Such distributed file systems include Google File System (GFS) (Ghemawat *et al.,* 2003), Hadoop Distributed File System (HDFS) (Hadoop, n.d.), Cloudstore (formerly Kosmos File System) (Cloudstore, n.d.), Sector (Gu & Grossman, 2009) and Ceph (Weil *et al.,* 2006).

Digital forensics in traditional computational environments is a subject thoroughly examined in the last decade. The procedures followed in order to gather digital evidence with ensured admissibility in court, are described in standard operating procedures documentation such as the ACPO guidelines (ACPO, 2008). In addition, a variety of forensics acquisition tools has been developed (e.g. the Forensic Toolkit (FTK), EnCase and Foremost) which can automate the collection and analysis – to some extent – of evidence.

However, data distribution and resource pooling in a cloud make the investigator's work much more challenging than in a traditional computational environment as existing digital forensics tools seem inappropriate.

In addition, every country is governed by its own privacy policies and laws. Thus, gathering digital evidence from a cloud's server that is located in a foreign country, outside of our jurisdiction area, could result in violating the country's privacy protection legislation (Taylor *et al.*,2010; Garrison *et al.,* 2010). Still, the legal procedure to gain access to evidence held in a public cloud may lead in acquiring wrong data and result into privacy violations. Grobauer and Schreck (2010) present incident handling issues in the cloud.

3. Challenges in Forensics Data Acquisition in Cloud Systems

A cloud system poses many obstacles to forensic data acquisition. These obstacles involve both technical setbacks and legal restrictions. The conjunction of these two factors can raise serious challenges in cloud forensics. A coarse classification is shown in Table 1.

The terminology used in this paper is clarified below:

- Live data mainly refer to the data that belong to the cloud's user/suspect and still have not been deleted (or have not been permanently deleted). In this point it is obvious that the user/suspect still uses the cloud.
- Permanently deleted data refer to the data that the suspect has permanently deleted.
- By jurisdiction we refer to whether we have the right to access the data (servers are inside jurisdiction) or not (servers are outside jurisdiction).

Table 1. Challenges

State of data / Jurisdiction	Live	Permanently Deleted
Within	(almost) traditional forensics	Metadata acquisition - Lack of knowledge about the data owner
Outside	Transfer data within jurisdiction area and treat as traditional forensics	Metadata acquisition - Privacy violation - Lack of knowledge about the data owner

In general, a distributed file system architecture comprises of a master control server (in many cloud distributions it is referred as namenode or master) and several storage servers (also known as datanodes or slaves). Masters maintain the metadata of the data that reside in the cloud and are also responsible for the delegation of data to slaves. Slaves are used to store the actual data.

4. GFS Architecture

As described earlier, like any distributed file system, GFS comprises of a master server, and multiple slaves, and is accessed by multiple users as shown in Figure 1.

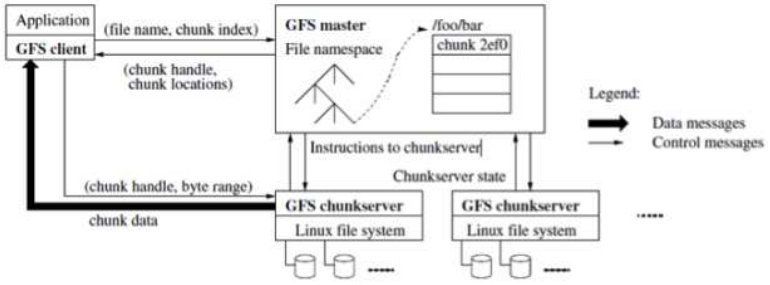


Figure 1. GFS structure (*source:* Ghemawat *et al.*, 2003).

According to Ghemawat *et al.* (2003), user files in a GFS are divided by the master into fixed-size chunks (Figure 1). Each chunk is identified by an immutable and globally unique 64-bit chunk handle assigned by the master at the time of chunk creation. Files and chunks are all uniquely and eternally identified by the logical times at which they were created.

Slaves (or chunkservers) do not store whole files, instead they store chunks on local disks, according to their local file system (e.g. ext3), and read or write chunk data

specified by a chunk handle and byte range. Furthermore, a file's chunks can be distributed in more than one slave. For fault tolerance, each chunk is replicated more than once (preferably three times), in different slaves. As a result, each replica is stored as a plain file in the slave's local file system.

The master maintains all file system metadata. This includes file and chunk namespaces, access control information (in sector/sphere access control information is stored in a different server), the mapping from files to chunks and the current location of the chunks. All metadata is kept in master's memory. However, namespaces, along with access control and mapping information are also kept persistent by logging mutations to an operation log stored on the master's local disk and replicated on remote machines. Chunk location is not stored persistently.

When a file is deleted, the master logs the deletion instantly and eventually, after an interval of time during which the file can be recovered in case of an accidental delete (this interval is three days, but can be configurable), all its metadata (both file's and chunks') are deleted from the master's memory. In addition, all its chunks and their replicas are also deleted from the slaves. After the deletion, the space that the chunks used to hold is considered as free.

5. Forensic acquisition scenarios

In the traditional IT infrastructure, it is relatively easy to gather evidence from digital storage devices. This is typically achieved by performing a bit-stream, exact copy of the suspect's hard drive. However in a cloud system the situation is more complicated as we cannot have direct access to the hardware and sometimes investigation can face legal impediments. Our methodology commences by developing four scenarios showing then challenges that arise when attempting to retrieve data from a cloud (Table 1). We then use these scenarios to draw the requirements and develop algorithms to enable forensic acquisition.

5.1. Scenario 1: Live Data - Within Jurisdiction

In the first scenario, we examine the situation in which the state of the suspect's data is still live (files have not been deleted) and the slaves where files' chunks are held are within jurisdiction area (Figure 2a.). In this situation there are not any particularly problems for the investigator to gather the data, as she can acquire them through the suspect's computer or just by having access to the master server. In the second case, the investigator must be able to recognise suspect's files among other users' files inside the master in order to maintain the privacy of the non-suspect users. After the distinction has been done, it is relatively straightforward to access those files. Well known standard operating procedures can be applied in this case for live acquisition similar to internet-based evidence gathering (Shipley, 2009a;2009b).

5.2. Scenario 2 Live data – Outside jurisdiction

In this scenario the data are still available to the end user, but unlike the first scenario the investigator cannot access either them or their copies. This implication comes when the location of the slaves that maintain the chunks of these data is out of our

jurisdiction area, perhaps in another country with different legislation. Thus, legal arrangements and agreements have to be done between involved countries for such situations. However, it would be much more preferable, in terms of timeliness and bureaucracy, if we had the possibility to avoid involvement of a foreign country.

In this case we can leverage the cloud's functionality to transfer the files, or more specifically the chunks of the file, within our jurisdiction area without manually obtaining them. An approach of transferring suspect's data within jurisdiction area is to activate the cloud's inherent mechanism of redundancy by introducing failures (Figure 2b.). When the master detects that one of the file's copies, or even the file itself, is unavailable due to a failure, it tries to make a new replica of it. The new replica's chunks are stored in a slave (or multiple slaves) according to the mechanism the cloud uses. Thus, by deliberately deleting one of the file's replicas or just by deleting its metadata in the master we can force the cloud to make another replica. Manipulating the cloud's mechanism in order to store the replica's chunks in a slave (or slaves) within our jurisdiction area we have achieved our goal. From this point on, data are considered to be within jurisdiction and as in Scenario 1 the investigator can easily obtain them.

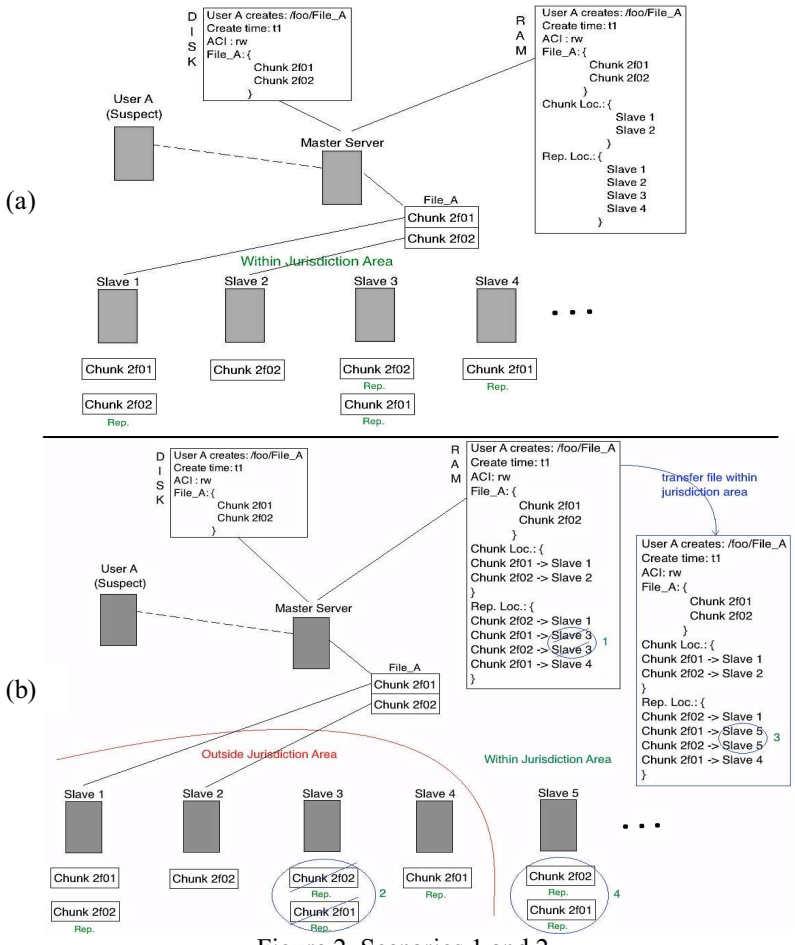


Figure 2. Scenarios 1 and 2

5.3. Scenario 3 Permanently Deleted – Within Jurisdiction

As we described above for the GFS, upon file deletion GFS does not immediately reclaim the available physical storage either in master's memory or is slaves' disks. That means that neither the metadata of the file nor the chunks that compose it are deleted. This happens to improve reliability in an event of an accidental delete. Thus, the file is not completely lost, it is just renamed to a hidden name and can be read under its new name or recovered simply by renaming it back to normal (this functionality is similar to the conventional operating systems recycling bin). This mechanism would be very helpful in the case of a digital forensics investigation. However, both metadata and chunks after a file's deletion do not remain in the system forever. More specifically, after the interval of three days has elapsed, the hidden file is removed from the namespace, its in-memory metadata are erased and slave's storage space, in which the deleted file's chunks used to reside, is freed. From this point on, the investigator is not capable of acquiring the deleted file directly.

At a first level, from a forensic readiness and compliance perspective certain regulations requiring deleted data to be retained for a period comparable to telecoms industry would partially address the problem. These regulations should be implemented in the existing legislation on digital data retention, such us European Union's Data Retention Directive (2006) and Electronic Evidence Compliance by the U.S. Internet Service Provider Association (2003).

However, in a cloud system the amount of data stored in it is significantly large and what is more it is constantly growing. Thus, retention of data can be a rather resource consuming and expensive process. As such, the period in which deleted data can be retained may not meet the expectations and needs of the investigator. Another way for deleted data acquisition is required.

After the expiration of the period during which a deleted file is preserved, the state of the deleted file and its chunks in the system is as follows:

A. From the master's perspective:
- File namespace: Stored permanently in master's disk in a log file
- Chunk namespace: Stored permanently in master's disk in a log file
- Mapping file to chunks: Stored permanently in master's disk in a log file
- Location of the chunks: Lost. Not logged.

B. From the slaves' perspective:
- Chunk deletion follows the normal deletion of a file in the slave's local file system, since every chunk is considered as a unique file.
- Deleted chunks retain their position on the disk as long as they are not overwritten by other chunks.
- Chunk metadata in the local file system are retained as long as they are not deleted by other metadata.

As it is shown, even though the file has been deleted from the master, along with its chunks from the slaves', not all of the information is lost. In fact, information that link chunks' names to a file is persistently preserved (Figure 3).

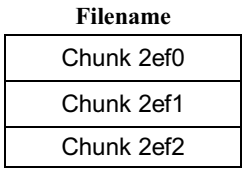


Figure 3.A File consisting of three chunks

In addition, these chunks that compose the deleted file along with their local metadata still exist somewhere in the cloud, at least as long as they are not overwritten by other chunks and metadata respectively. Moreover, the fact that multiple copies of the file's chunks exist, the possibility for them to be found before they get overwritten can be non-negligent. However, unlike common file systems it is not easy to determine whether these chunks belong to the suspect's deleted file. A slave server accommodates data, which belong to more than one user. Furthermore, we do not have any indication on where to look for these chunks, since the location of each chunk is not logged by the master. Thus trying to harvest data from a crop of slaves might lead to privacy issues. As such, a straightforward dd application on the slave would not be a suitable, legal option.

For example, let us suppose that user_A (our suspect) creates a file (File_A), which is comprised of chunk_1234, and chunk_1235 located on slave 1 and slave 2 respectively. Their replicas are distributed into the slaves as shown in Figure 4a.

Later on the suspect deletes her file. As we can see in Figure 4b, all respective metadata in master's volatile memory are destroyed. Also the space in the slaves' local file systems where the chunks and their metadata resided is now considered free and ready to be used by other data.

At this point, although the suspect has deleted her file, its contents still exist in the cloud, but the location is not known since the master does not keep any information about the respective chunk locations. A brute force search approach involving every slave, using their names as a keyword would require excessive amount of time and resources. Therefore a first requirement emerging from this scenario would be the need for the master to store the chunk locations on its disk in the form of a log file, as it does with the other metadata. By maintaining a location history log file the recovery would be considerably more efficient than exhaustive search, whenever of course recovery would be possible.

In the meantime another user, userB, uses the cloud to store his data (let us say that he wants to store file_B). As he creates the file in the cloud and chunks are being created, three distinct alternatives can take place, as it is shown in Figure 5a:

1. Chunks that belong to file_B can overwrite chunks that belonged to the deleted file_A.
2. Metadata of the chunks that belong to file_B can overwrite metadata of the chunks that belonged to file_A.

3. Chunks of file_B and their metadata leave chunks of file_A and their metadata unaffected.

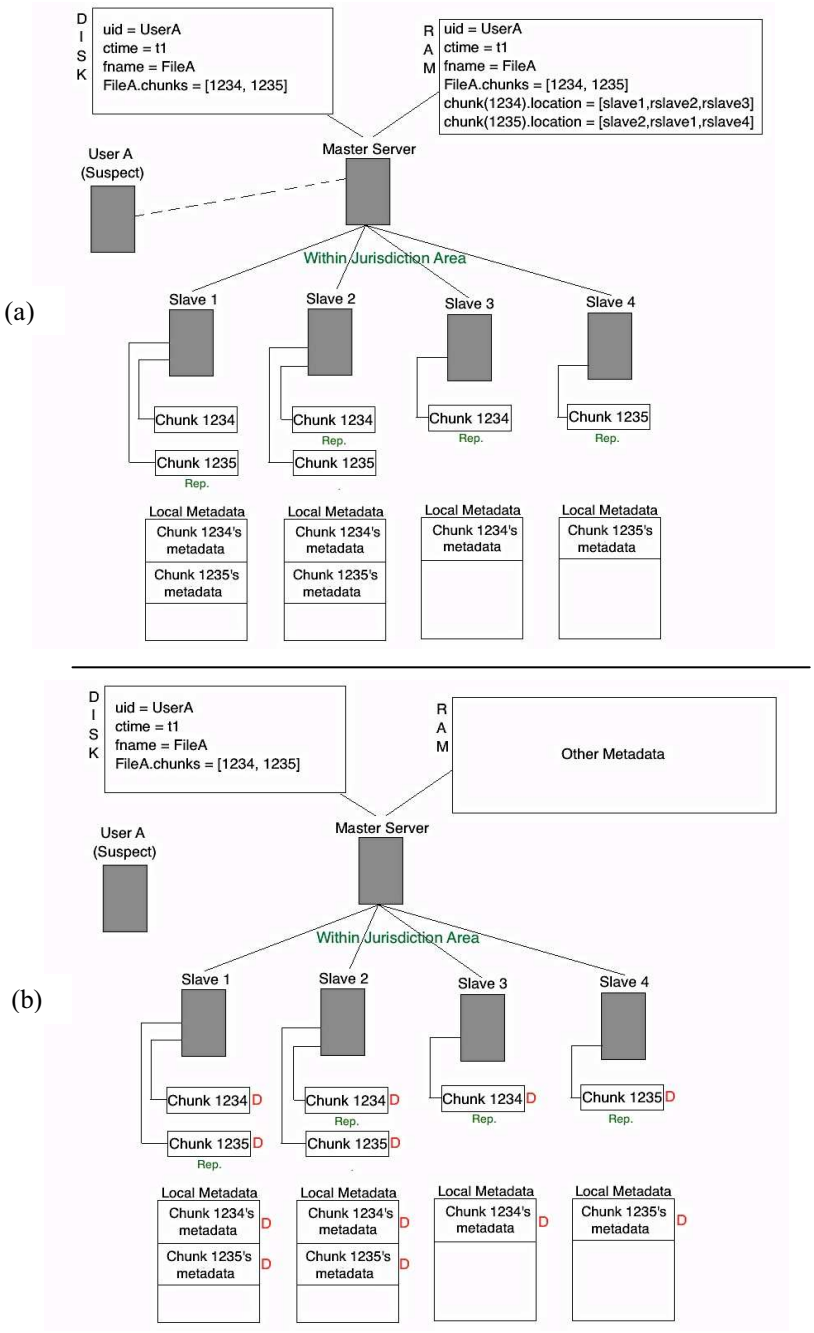


Figure 4. File creation (a) and deletion (b)

According to the first alternative, Chunk_1236 of file_B overwrites Chunk_1235. However, metadata of Chunk_1235, though deleted they still exist. This is due to

the slave's local file system (for example ext3). Acquisition of File_A chunks would be performed by reading the deleted metadata from the local file system of the slave (slave1) as indicated by the master's metadata (since according to our proposed approach the chunk locations are also stored on the master's disk), followed by examining the block bitmap for the blocks that are indicated by the metadata, only to discover that those blocks are allocated. From this it is concluded that another chunk has overwritten the chunk of file_A.

In the second alternative, the metadata of Chunk_1237 of file_B overwrite those of Chunk_1235 of file_A. In this case, Chunk_1235 will continue to exist in slave 4, but there will be no way to find it, because its metadata is overwritten.

For the third alternative there is no complication, as the old chunks and metadata coexist with the new ones.

Lastly, userB decides to delete his file (Figure 5b). At this point, the alternatives described above become even more challenging. In the first alternative both deleted metadata of Chunk_1235 and Chunk_1236 point at the same position in the slave. Unlike before, now we cannot check the block bitmap to determine if this chunk belongs to the suspect's file_A. At a fisrt stage we cloud check the chunk deletion times from their metadata in the local file system and decide which one of them was the last chunk written in this particular space of the disk. However, if the deleted metadata of Chunk_1236 were overwritten by other metadata, only metadata of Chunk_1235 would survive pointing at a position on the disk that Chunk_1236 resides. This could eventually lead to gathering wrong evidence and violating the privacy of userB.

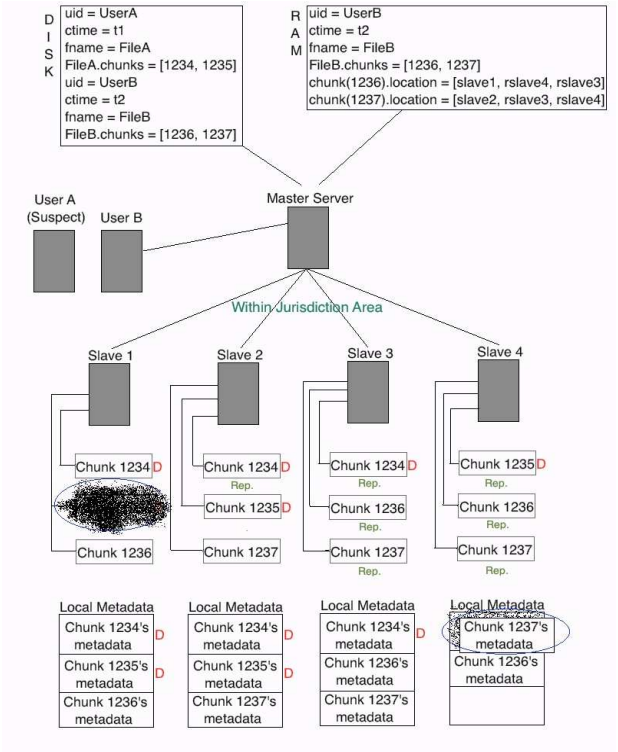
In order to present an accurate reflection of file deletion dynamics in a distributed file system we need to highlight the fact that chunk consists of many blocks and it is not likely for a chunk to be completely overwritten by another chunk. Therefore we should rather talk about chunk blocks being overwritten rather than chunks.

Against the above we are driven to the conclusion that evidence acquisition in a cloud is mostly based on chunk blocks. For this reason, in order to examine each block of a chunk separately there should be a "block to last chunk's name" mapping mechanism where each block should be connected with the name of the chunk that it lastly belonged to.

5.4. Scenario 4 Permanently Deleted – Outside jurisdiction

In this scenario not only the data are permanently deleted, and the investigator cannot obtain them directly (if they still exist somewhere in the cloud), but the deleted data are also out of the investigator's jurisdiction area. This is a major challenge for the investigator as the process of gathering data is the most complicated of all scenarios.

(a)



(b)

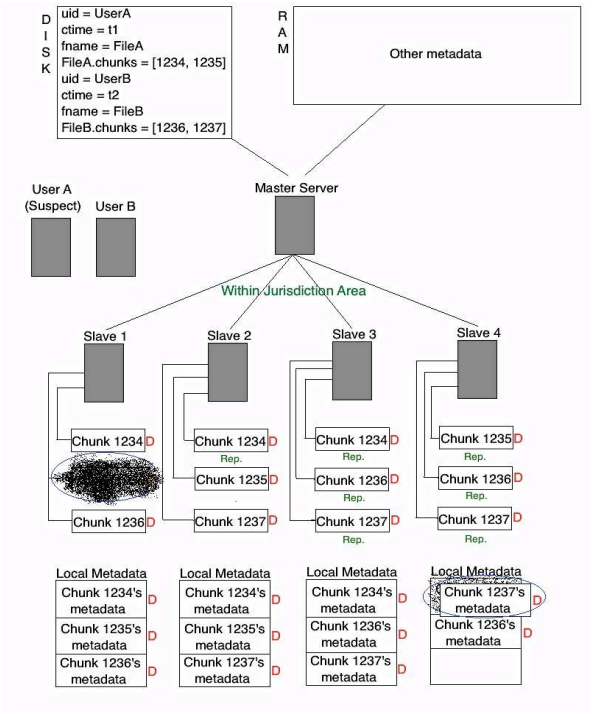


Figure 5. userA and userB delete their respective files

In this scenario, the acquisition of evidence cannot be conducted unless a legal agreement (such as the Mutual Legal Assistance Treaty) has been established between involved countries. This comes as a result to the fact that the cloud's inherent mechanism of redundancy does not work for deleted files.

Similar to the third scenario, using the "block to last chunk's name" mapping mechanism we introduced, we can easily determine which blocks belong to the chunks that compose the suspect's file(s). At this point, in order to simplify the work that needs to be done by the other countries where slaves are located, we formally request the respective parties through a mutual assistance treaty to "resurrect" (undelete) these blocks, by setting their bits in the block bitmap, under the chunk's metadata. Note that because some of the chunk blocks may have had been overwritten by another user's chunk blocks, the chunk may now be smaller than before (this can be checked by examining the "block to last chunk's name" map we previously introduced). If so, modifications to the chunk's metadata should be performed to reflect the smaller chunk length in order to partially recover the chunk (i.e. the part that belongs to the suspect user). In the meantime, we copy the metadata regarding the file from the master's disk into the master's memory. Eventually, we can recover the whole file, or part of it. Bringing those files within the area of our jurisdiction involves the procedures described in the second scenario.

5.5. Gap analysis summary

Summarising what we have introduced in the above scenarios in order a cloud storage system to be forensically ready the following gaps need to be filled:

1. Legal arrangements and agreements have to be established between involved countries in order to avoid illegal privacy violations.
2. Regulations regarding cloud's digital data retention when they are deleted should be implemented in the current related legislation.
3. From a technical point of view, the "block to last chunk name" mapping mechanism should be implemented in the cloud's structure.
4. Lastly the master server should persistently store the chunk locations along with the rest metadata described earlier on disk, in file records.

5.6. Requirements for a cloud forensics acquisition tool

NIST's Digital Data Acquisition Tool Specification (NIST, 2004) defines requirements for digital media acquisition tools used in computer forensics investigations. According to this specification and taking into account the characteristics of a cloud system, acquisition tools for cloud forensics should meet the following requirements:

- The tool shall be able to acquire a digital source using access to the master server and the suspect's machine, the only access interfaces visible to the tool.

- The tool shall be able to cause the activation of the cloud's inherent feature of redundancy.
- The tool shall be able to create either a clone of a digital source, or an image of a digital source, or provide the capability for the user to select and then create either a clone or an image of a digital source.
- The tool shall operate in at least one execution environment and shall be able to acquire digital sources regardless to the user's, master's or slaves' local execution environment.
- The tool shall completely acquire all visible data sectors from the indicated by the master's metadata area in the slaves.
- The tool shall completely acquire all hidden data sectors from the indicated by the master's metadata area in the slaves.
- All data sectors acquired by the tool from the slaves shall be accurately acquired.
- If there are unresolved errors reading from a slave then the tool shall notify the user of the error type and the error location.
- If there are unresolved errors reading from a slave then the tool shall use a benign fill in the destination object in place of the inaccessible data.

The tool, except from mandatory features described above, may also offer additional features, according to the NIST specification.

6. Concluding remarks

Despite the advantages the introduction of a cloud infrastructure may bring into a corporate network, the additional layer of complexity will create further challenges and issues that the first responder and forensic analyst will need to address during the digital evidence acquisition phase. In this paper we considered a coarse distinction between deleted and undeleted data as well as within and outside jurisdiction. The need for revising the technical, procedural and legal requirements was realised. Forensic readiness should not be an afterthought as a security incident involving litigation is likely to become commonplace. As such, a forensically ready cloud storage is important for the sustainability, success and adoption of cloud services.

7. References

ACPO, "Good Practice Guide for Computer-Based Electronic Evidence", 2008

Cloudstore, http://kosmosfs.sourceforge.net

EC, "Directive 2006/24/ec of the european parliament and of the council of 15 March 2006 on the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks and amending Directive 2002/58/EC", Official Journal of the European Union, 13/04/2006.

"Electronic Evidence Compliance – A Guide for Internet Service Providers", 18 Berkeley Tech. L.J. 945, 947 (2003) (Prepared by the U.S. Internet Service Provider Association).

Garrison, C, Schiller, C., Steele, J. “Digital Forensics for Network, Internet, and Cloud Computing”, Syngress Media, 2010

Ghemawat, S. Gobioff, H., Leung, S. “The Google file system”, 19th ACM symposium on Operating systems principles, October 19-22, 2003, Bolton Landing, NY, USA

Grobauer, B. and Schreck, T., “Towards Incident Handling in the Cloud: Challenges and Approaches”, Proceedings of the 2010 ACM workshop on Cloud computing security workshop.

Grobler, T. and Louwrens, B., 2007, in IFIP International Federation for Information Processing, Volume 232, New Approaches for Security, Privacy and Trust in Complex Environments, eds. Venter, H., Eloff, M., Labuschagne, L., Eloff, von Solms. R., (Boston: Springer), pp. 13–24.

Gu, Y, and Grossman, R. “Sector and Sphere: the design and implementation of a high-performance data cloud”, Phil Trans R Soc A 367, 2009, pp. 2429-2445.

Hadoop File System, http://hadoop.apache.org/hdfs

Mell, P. and Grance, T. “The NIST Definition of Cloud Computing”, National Institute of Standards and Technology, 2009.

National Institute of Standards and Technology, “Digital Data Acquisition Tool Specification”, 2004.

Patzakis J., Computer Forensics as an Integral Component of Information Security Enterprise, Guidance Software, Available from: http://www1.stpt.usf.edu/gkearns/Articles_Fraud/computerforensics.pdf, 2003

Shipley, T. “Collection of evidence from the Internet: Part 1. A basic methodology”, Digital Forensic Investigator, http://www.dfinews.com/article/ collection-evidence-internet-part-1, 2009a

Shipley, T. “Collection of evidence from the Internet, Part 2: The cloud”, Digital Forensic Investigator, http://www.dfinews.com/article/collection-evidence-internet-part-2, 2009b.

Taylor, M., Haggerty, J., Gresty, D., Hegarty, R. “Digital evidence in cloud computing systems”, Computer Law & Security Review, Volume 26, Issue 3, 2010, pp. 304-308

Thanh, T. Mohan, S. Choi, E. Kim, S., Kim, P. "A Taxonomy and Survey on Distributed File Systems," Fourth International Conference on Networked Computing and Advanced Information Management, vol. 1, pp.144-149, 2008

Weil, S, Brandt, S., Miller, E. Long, D., Maltzahn, C. “Ceph: a scalable, high-performance distributed file system”, 7th symposium on Operating systems design and implementation, November 06-08, 2006, Seattle, Washington

8. Appendix

Algorithm 1: Transfer live files inside jurisdiction area

Input: user's filenames
Output: user's filenames

```
B={b_j | b_j=(fname, state, chunks[],
      chunks.location[], ...)}
For j=1..|B|
    If b_j.state == live
        If chunks.location[] != wslave[]
            // wslave[] = slaves within jurisdiction
              For k=1..|chunks|
                   del chunks[k].location[n]
                // n = a number between 1 and m
        // m = number of a chunk's replicas + 1
              End For
              create new replica in wslave[]
        End If
    End if
End for
```

Algorithm 2: Acquire permanently deleted, inside jurisdiction area files through master

Input: user's filenames
Output: user's files

```
B={bj | bj=(fname, state, chunks[],
      chunks.location[], …)}
For j=1..|B|
  Acquire chunks.location.metadata[]
  For each unique chunk
    For k=1..|chunks.location.metadata.blocks[]|
       Check block to last chunk's name map
        If last chunk's name != fname.chunk[of
                                    unique chunk]
            break
        else
          Acquire block(k)
          Chunkfile[k] = block(k)
        End if
      End for
       File = File + Chunkfile[of unique chunk]
      End for
End for
```