Towards Building Proofs of Past Data Possession in Cloud Forensics

Shams Zawoad
University of Alabama at Birmingham
Birmingham, Alabama 35294-1170
Email: zawoad@cis.uab.edu

Ragib Hasan
University of Alabama at Birmingham
Birmingham, Alabama 35294-1170
Email: ragib@cis.uab.edu

ABSTRACT

Cloud computing has emerged as a popular computing paradigm in recent years. However, today's cloud computing architectures often lack support for computer forensic investigations. A key task of digital forensics is to prove the presence of a particular file in a given storage system. Unfortunately, it is very hard to do so in a cloud given the black-box nature of clouds and the multi-tenant cloud models where many users share the same storage and the content of the storage changes frequently. In clouds, analyzing the data from a virtual machine instance or data stored in a cloud storage only allows us to investigate the current content of the cloud storage, but not the previous contents. In this paper, we introduce the idea of building proofs of past data possession in the context of a cloud storage service. We present a scheme for creating such proofs and evaluate its performance in a real cloud provider. We also discuss how this proof of past data possession can be used effectively in cloud forensics.

I INTRODUCTION

In the recent years, cloud computing has gained popularity as a high-performance and low-cost computing paradigm, which provides better utilization of resources using virtualization. Small and medium scale companies find cloud computing highly cost effective as it replaces the need of building costly physical and administrative infrastructure, and offers the pay-as-you-go structure for payment. Khajeh-Hosseini et al. found that an organization could save 37% cost if they would migrate their IT infrastructure from an outsourced data-centre to the Amazon Cloud [1]. Recent research by Market Research Media states that the global cloud computing market is expected to grow at an 30% Compound Annual Growth Rate (CAGR) reaching $270 billion in 2020 [2]. According to Gartner Inc., the strong growth of cloud computing will bring $148.8 billion revenue by 2014 [3]. Cloud computing is getting popular not only in the private industry, but also in the government sector. According to a research from INPUT, the federal cloud market is expected to expand to $800 million by 2013 [4].

However, the characteristics of cloud computing not only attract business organizations, but also attract malicious individuals who can use clouds to evade the law. There are two dimensions of attacks in cloud computing – an attacker can target an application deployed in cloud, or she can use the cloud to launch an attack. This new attack surface has opened a new area in digital forensics – *Cloud Forensics.* According to an annual report of Federal Bureau of Investigation (FBI), the size of the average digital forensic case is growing at 35% per year in the United States. From 2003 to 2007, it increased from 83GB to 277 GB in 2007 [5]. This rapid increase in digital forensics evidence drove the forensic experts to devise new techniques for digital forensics. At present, there are several established, proven digital forensics tools in the market. Unfortunately, because of the characteristics of cloud computing, many of the assumptions of digital forensics are not valid in the cloud computing model, e.g., in a cloud environment, we do not have physical access to the evidence, which we take for granted in traditional privately and locally owned computing systems. Hence, cloud forensics brings new challenges from both technical and legal point of view and opens a new research area for security and forensics experts.

The process of digital forensics starts with acquiring the digital evidence. In a cloud, the evidence could be the image of virtual machine, files stored in cloud storage, and logs provided by cloud service providers (CSP). If an adversary shuts down the virtual machine or removes files from cloud storage, then there is no way to prove what file she possessed previously. To overcome this problem, Birk et al. proposed continuous synchronization [6]. However, there is no solution which states how to do the continuous synchronization and how to prove past data possession.

In this paper, we take the first step towards building a proof of past data possession (PPDP). Such a proof can be used by forensic investigators to prove that a suspect stored a given file with a service provider at a past time period.

To illustrate the problem, we present the following hypothetical scenario:

Alice got access to some illegal insider trading [7] information of XYZ corporation. Alice used this non-public information to gain illegal advantage in trading of stock and bonds. She hosted the file in a cloud storage and removed the file after trading to hide her tracks. After repeating this incident for several times, she got the attention of law enforcement agency and they assigned Bob to investigate the case. Bob acquired the possible insider trading information from XYZ corporation but could not prove whether Alice had access to and stored those files, as she deleted those after trading. If Alice used her personal computer to store the information, then the task for Bob would be easy. Using some current forensic tools, he could retrieve the deleted files or prove their existence in Alice's computer. But as Alice used the cloud storage, there was no way for Bob to prove the data possession. Cloud provider could store all the deleted files, but this would increase the storage cost extensively. Moreover, if cloud provider published the proof of the files insecurely, an intruder, Charley could get information of the files possessed by Alice. On the other hand, Alice might claim that Bob colluded with the cloud provider to prove a false allegation.

To mitigate the challenges discussed in the above scenario, we propose the notion of a proof of *past data possession* in this paper.

Contributions: The contributions of this paper are as follows:

1. We introduce the notion of a Proof of Past Data Possession (PPDP) in the context of digital forensics;
2. We propose an efficient and secured cryptographic scheme for creating a PPDP; and
3. We evaluate the proposed PPDP scheme using a commercial cloud vendor.

Organization: The rest of this paper is organized as follows. Section II provides some background informations and challenges of cloud forensics. Section III describes the adversaries capabilities and possible attacks on past data possession. Section IV presents our scheme for creating a proof of past data possession, security analysis of the scheme, and performance evaluation in a real cloud environment. Section V discusses the applicability of such proofs in cloud forensics and in regulatory compliance. In Section VI, we provide an overview of related research in cloud forensics, and finally, we conclude in Section VII.

II BACKGROUND AND CHALLENGES

With the incrusting popularity of cloud computing, there is a significant interest in the law-enforcement community to extend digital forensics techniques in the context of a cloud. In this section, we present the definitions of digital forensics and cloud forensics, motivation of our work, and discuss about the challenges of cloud forensics.

1 DIGITAL FORENSICS

Digital forensics is the process of preserving, collecting, confirming, identifying, analyzing, recording, and presenting crime scene information. Wolfe defines digital forensics as *"A methodical series of techniques and procedures for gathering evidence, from computing equipment and various storage devices and digital media, that can be presented in a court of law in a coherent and meaningful format"* [8]. According to a definition by NIST [9], computer forensic is an applied science to identify a incident, collection, examination, and analysis of evidence data. While doing so, maintaining the integrity of the information and strict chain of custody for the data is mandatory. Several other researchers define computer forensics as the procedure of examining computer system to determine potential legal evidence [10, 11].

2 CLOUD FORENSICS

Cloud forensics can be defined as applying computer forensics procedures in cloud computing environment. As cloud computing is based on extensive network access, and as network forensics handles forensic investigation in private and public network, Ruan et al. defined cloud forensics as a subset of network forensics [12]. They also identified three dimensions in cloud forensics – technical, organizational, and legal. Cloud forensics procedures will vary according to the service and deployment model of cloud computing. For Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS), we have very limited control over process or network monitoring. Whereas, we can gain more control in Infrastructure-as-a-Service (IaaS) and can deploy some forensic friendly logging mechanism. The first three steps of computer forensics will vary for different service and deployment model. For example, the evidence collection procedure of SaaS and IaaS will not be same. For SaaS, we solely depend on the CSP to get the application log, while in IaaS, we can acquire the Virtual machine instance from the customer and can enter into examination and analysis phase. On the other hand, in private deployment model, we have physical access to the digital evidence, but we merely can get physical access to public deployment model.

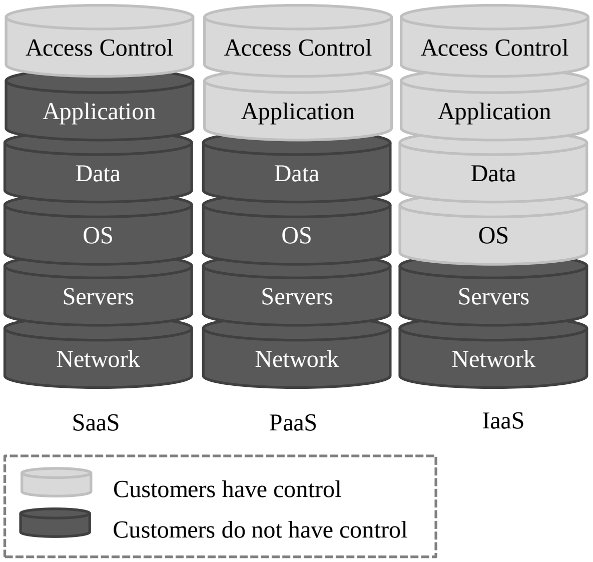

Figure 1: Customers' control over different layers in different service model.

3 MOTIVATION

Though cloud computing offers numerous opportunities to different level of consumers, many security issues of cloud environment have not been resolved yet. According to a recent IDCI survey, 74% of IT executives and CIO's referred security as the main reason to prevent their migration to the cloud services model [13]. Some recent and well-publicized attacks on cloud computing platform justify the concern with security. For example, a botnet attack on Amazon's cloud infrastructure was reported in 2009 [14].

Besides attacking cloud infrastructure, adversaries can use the cloud to launch attack on other systems. For example, an adversary can rent hundreds of virtual machines to launch a Distributed Denial of Service (DDoS) attack. After a successful attack, she can erase all the traces of the attack by turning off the virtual machines. A criminal can also keep her secret files (e.g., child pornography, terrorist documents) in cloud storage and can destroy all her local evidence to remain clean. When law enforcement investigates such a suspect, the suspect can deny having the illegal content in her cloud storage. At present, there is no way to claim that an adversary owns certain data at a given time.

Researchers are working to protect the cloud environment from different types of attacks. But in case of an attack, we also need to investigate the case, i.e., we need to carry out a digital forensic investigation in the cloud. Besides protecting the cloud, we need to focus on this issue. Unfortunately, there have been little research on adapting digital forensics for use in cloud environments. In this paper, we address this problem, which has significant real-life implications in law enforcement investigating cybercrime and terrorism.

4 CHALLENGES

The inherent characteristics of a cloud have made cloud forensics a nightmare for the forensic investigators. It is very difficult to prove the possession of data at a past time period. For example, how can an investigator prove that the suspect stored an incriminating file last month in the cloud, a file she has recently deleted from the cloud storage? We must find secure techniques for creating such proofs of past data possession which will be admissible in a court of law as valid evidence. Many things can complicate such a proof. Clients may question the integrity of any such proofs, claiming that the forensic investigators or the prosecution and the CSP have colluded to plant an evidence in the cloud or have simply lied about the presence of the incriminating files in the cloud. The following reasons also make the proof of past data possession challenging.

Less Control in Clouds: One of the challenges to prove the past data possession is the less control in clouds. In traditional computer forensics, the investigators have full control over the evidence (e.g., a hard drive confiscated by police). In a cloud, unfortunately, the control over data varies in different service models. Figure [1]shows the control of customers in different layers for the three different service models – IaaS, PaaS, and SaaS. From the figure, we can observe that consumers have highest control in IaaS and least control in SaaS. This physical inaccessibility of the evidence and lack of control over the system make evidence acquisition a challenging task in cloud forensics. For example, in our hypothetical case, Bob could have easily used traditional forensics tools to recover the deleted files if he had the physical access to cloud storage. Sometimes, it is even impossible to locate where the data actually reside physically. Several other problems come along with this issue. For example, we need to depend on the cloud service providers (CSP) for evidence acquisition. Which in turn brings the honesty issue of the CSP's employee, who is not a certified forensic investigator. Additionally, CSPs are not always obligated to provide all the necessary logs.

Multi-tenancy: In cloud computing, multiple virtual machines (VM) can share the same physical infrastructure, i.e., data for multiple customers may be co-located. This nature is different from the traditional single owner computer system. Hence, while generating the proof of data possession for one user, other users' data can be mingled with the proof. An alleged user can claim that the proof contains information of other user, not her. The cloud provider then needs to prove it to the court that the proof indeed contains the information of the malicious user. Moreover, we need to preserve the privacy of the other tenants. For both of these issues, preserving the proof of past data possession becomes more challenging.

Chain of custody: Chain of custody is one of the most vital issues in traditional digital forensic investigation. Chain of custody should clearly depict how the evidence was collected, analyzed, and preserved in order to be presented as admissible evidence in court [15]. In traditional forensic procedure, it starts with gaining the physical control of the evidence, e.g., computer, hard disk. However, in cloud forensics, this step is not possible due to the multi jurisdictional laws, procedures, and proprietary technology in cloud environment [16, 17]. The proof of past data possession needs to follow this chain of custody. We must need to clarify certain things to maintain the chain of custody, e.g., how the proof was generated, how it was stored, and who had the access to these proofs.

Presentation: The final step of digital forensic investigation is presentation, where an investigator accumulates his findings and presents to the court as the evidence of a case. Challenges also lie in this step of cloud forensic [18]. Proving the evidence in front of jury for traditional computer forensic is relatively easy, compared to the complex structure of cloud computing. If an investigator retrieves some deleted files from a personal computer, it will be easily presentable to the court. As we cannot retrieve the deleted files from the cloud storage, the proof of data possession must be presented to the court in a convenient way.

III THREAT MODEL

In this section, we describe the attacker's capability, possible attacks on data possession, and properties of our proposed system. Before describing the threat model, we first define the important terms to clarify the threat model.

1 DEFINITION OF TERMS

- *User:* A user is a customer of the cloud service provider (CSP), who uses the CSP's storage service. A user can be malicious or honest.

- *Evidence:* An evidence can be a file, virtual machine image, log files, and any other digitally acquired data.

- *Proof of Past Data Possession (PPDP):* The PPDP contains the proof of data possession to verify whether the user actually possessed the evidence or not.

- *CSP:* The Cloud Service Provider (CSP) will generate the PPDP and provide an web interface to users and investigators for verifying the evidence.

- *Investigator:* An investigator is a professional forensic investigator, who needs to verify the evidence from PPDP. An investigator can also be malicious or honest.

- *Intruder:* An intruder can be any malicious person, who wants to reveal some information of the evidence from the proof of past data possession.

2 ATTACKER'S CAPABILITY

In our threat model, we assume that the users and the investigators do not trust the CSP, and both of them can be malicious. A user can delete records from her storage but cannot change the PPDP by herself. An investigator can plant a false evidence only when he colludes with the CSP, and cannot change the PPDP by himself. A CSP can produce false PPDP only after colluding with a user or an investigator. The CSP can deny hosting any evidence or repudiate any published PPDP. An intruder can acquire the PPDPs of user to learn about the evidence or evidence change history from the proofs.

3 POSSIBLE ATTACKS

There can be different types of attacks on past data possession. A user can deny any evidence ownership, an investigator can produce invalid proof, even a CSP can deny hosting a file. Below we mention the possible attacks:

- *Denial of possession:* A user can delete file from her cloud storage. Later, if an investigator found other evidence of her deleted file, she denies to have the file. In a variation of this, a colluding investigator can also delete evidence from the user's storage and deny to find the evidence.
- *False presence:* If an investigator is not trustworthy, he can plant a false evidence. A user can also present a false proof, which can make her free from the accusation.
- *Evidence contamination:* User and investigator can modify the evidence to prove their claim.
- *Repudiation by CSP:* An otherwise honest CSP can deny hosting a file or can deny a published PPDP after-the-fact.
- *Repudiation by User:* As data are co-mingled in the cloud, a malicious user can claim that the published proof contains other cloud user's data.
- *Privacy violation:* As the CSP published the proof of past data possession publicly on the web, any malicious person can acquire the published PPDP and try to learn about the evidence from the proof. This attack can violates the privacy of user data hosted in the cloud.

4 SYSTEM PROPERTY

Our mechanism should prevent any malicious party to produce a false proof of past data possession. A false past data possession proof attests the user's possession of record, which the user does not actually own. Once the proof has been published, the CSP can neither modify the proof nor repudiate any published proof. We must also prevent false implications by dishonest forensic investigators. Our system provides the following integrity and confidentiality properties:

I1: An investigator or user whether acting alone or colluding together cannot remove any evidence.

I2: Two colluding investigator and user, or investigator and user acting alone cannot plant any invalid evidence.

I3: An investigator or user whether acting alone or colluding together cannot change any existing evidence.

I4: After publishing the proof of evidence, the CSP cannot deny hosting any evidence.

I5: The CSP cannot repudiate any previously published proof.

C1: From the published proof, no adversaries can recover any evidence.

C2: Adversaries will not be able to learn about the change history from the published proof.

IV PROOF OF PAST DATA POSSESSION

In this section, we present the proof of past data possession scheme, implementation, evaluation, and security analysis of our scheme.

1 THE SCHEME

We propose our scheme based on Bloom filters. Content of the Bloom filter is the proof of past data possession. A Bloom filter is a probabilistic data structure with no false negatives rate, which is used to check whether an element is a member of a set or not [19]. Bloom filter stores the membership information in a bit array. Bloom filters decrease the element insertion time and membership checking time. The only drawback of the Bloom filter is the probability of finding false positives. However, we can decrease the false positive probability by using a large bit array. In our scheme, we maintain separate Bloom filters for each users.

Proof Insertion: Figure [2] shows the flow of PPDP generation and below is the description of the process flow.

(a) A user can upload a file to a cloud server, create a new file, or a new log file can be generated in cloud environment.

(b) After acquiring any new or updated file, the CSP will get the last Bloom filter of that user, which is actually the bit array positions for all the previously inserted records of that user.

(c) The CSP will then hash the file including the user's information, generate the bit positions from the hash value, and update the bloom filter with the latest bit positions.

(d) Next, the CSP will store the updated bloom filter in the bloom filter storage.

(e) At the end of each day, the CSP will retrieve the Bloom filter entry of each user. We denote this as DS_u.

(f) Next, the CSP will hash the DS_u and sign it with its private key. This will be the $PPDP_u$ of that day.

$$PPDP_{\mathrm{u}} = <H(DS_{\mathrm{u}}), S_{\mathrm{PKC}}(H(DS_{\mathrm{u}})), t> \tag{1}$$

Where t is the proof generation time, *H(*DS_u*)* represents the hash of DS_u, and $\mathrm{S_{PKC}(H(DS_u))}$ represents the signature on the *H(*DS_u*)* using the private key of the CSP, *PKC*.

(g) After computing the proof of data possession, the CSP will publish the $PPDP_u$ and its public key PK on the web. These informations can also be available by RSS feed to protect it from manipulation by the CSP. The CSP will keep the DS_u and PKC secret to itself.

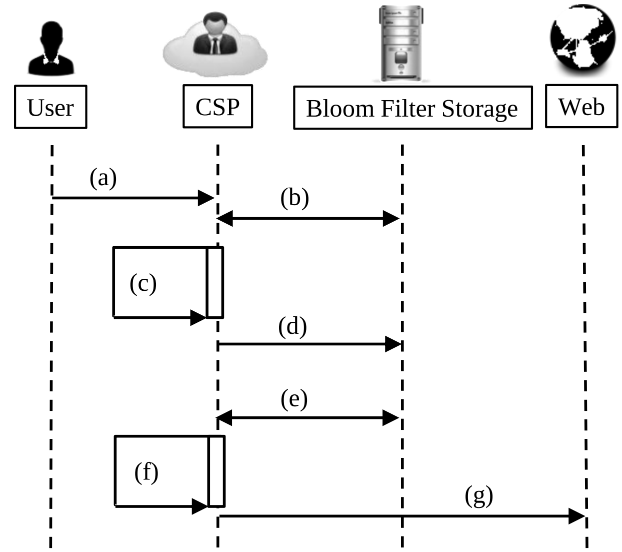

Figure 2: PPDP Generation Process Flow

A more advanced version of the above process can be used to prevent the repudiation by user attack for multi-tenancy case. As multiple users' data are co-located in same infrastructure, a user can claim that the proof contains information of her neighbour's file. To prevent this claim, all the evidence can be signed by the user's private key. Instead of hashing the file, the CSP now hashes the signature of the file to get the bit positions.

The false positive probability is proportional to the number of elements contained in the Bloom filter. There will be a change in the flow when the total number of evidence crosses the expected number of elements for the Bloom filter. If the expected number of elements is n, total number of bits required is m, and number of hash functions is k then the false positive probability p is determined by the following equation [20].

$$p = (1 - e^{\frac{-k*n}{m}})^k \tag{2}$$

According to the above equation, if the number of elements increases, the false positive probability will also increase. For example, if we configure the Bloom filter with a false probability of 0.01 for 1000 elements, then when the total number of evidence crosses 1000, the false positive probability will increase. Eventually, increasing false positive probability will increase the chance of planting false evidence attack. There are two options to mitigate this problem.

- *Option A:* After acquiring a file, the CSP will always check whether the current number of elements crosses the expected number of elements for the new file. If it does not cross the limit, then we can continue from step (b) of the above process flow. On the other hand, if it crosses the limit, then without getting the last bloom filter of the user, the CSP will create a new empty Bloom filter for the user and continue from step (c) using the newly created Bloom filter.

- *Option B:* As the above option add an extra checking before each proof insertion, it will increase the overhead of proof insertion. Another option can be creating a new Bloom filter of a user after a certain time. If a user creates w number of files on average in every month then we can create a new Bloom filter for her after every $\frac{n}{w}$ months. Using a fixed amount of time for creating new empty Bloom filter can generate two problems – for a user with low w, the space will be misused and for a user with high w, the false positive probability will increase. So creating a new Bloom filter after every $\frac{n}{w}$ months can balance the space requirement and the false positive probability.

Verification: Figure [3] shows the verification process flow and below is the details of the process.

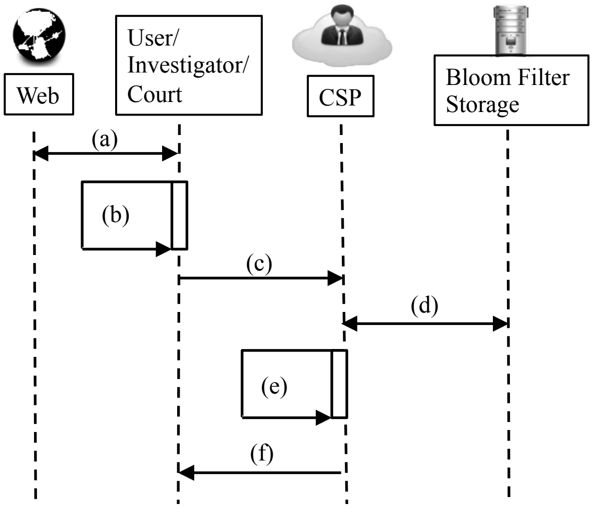

Figure 3: Evidence Verification Process Flow

(a) User, investigator, or court – any party can verify the validity of evidence. In the first step, they take the published $PPDP_u$ from the web.

(b) The verifying party will then check the validity of the $PPDP_u$. To check the validity, they first decrypt the signature of the hashed bloom filter by using the public key of the CSP. If the decrypted value and the hashed bloom filter are same, then they will proceed to next step.

(c) Next, the verifying party will upload the file with the user information from the management panel of the CSP.

(d) The CSP will then pick the last bloom filter entry, the hashed value of which was published on the web.

(e) Next, the CSP will hash the uploaded evidence with the user information, calculate the bits positions, and compare these bit positions with the retrieved Bloom filter entry. If all the calculated bit positions are set in the latest Bloom filter entry, we can say that the user actually possessed the file. One single false bit position means the evidence is not valid.

(f) After comparison, the CSP reports a positive or negative matching result to the verifying party.

Evidence Time-line Analysis: As the CSP publishes the PPDP everyday, we can identify the generation time of evidence. We can either identify the exact generation time or a time range in which an evidence was present.

To identify the generation time of an evidence, investigator will first send the evidence and user information to the CSP. The CSP will then acquire all the Bloom filter of the user and start searching to get a positive match. Positive match can be found for multiple PPDPs, but the generation time of the oldest PPDP will be the generation time for the evidence.

Investigator or court may also want to know whether a evidence was present in a particular time range. In that case, they will send the evidence, user information, and the desired time range to the CSP. After acquiring all the necessary informations, the CSP will retrieve the Bloom filters of that user, which lie in the particular date range. The CSP then starts a binary search to find a positive match. If a positive match is found, then we can say that the evidence was present in the selected time range.

2 SECURITY ANALYSIS

In our collusion model, there are three entities involved – CSP, user, and Investigator. All of them can be malicious individually or can collude with each other. We denote an honest CSP as C, a dishonest CSP as $\bar{C}$, an honest user as U, a dishonest user as $\bar{U}$, an honest investigator as I, and a dishonest investigator as $\bar{I}$. Hence, there can be total eight possible combination. Table [1] shows all the possible combinations.

CSP Honest?	**User Honest?**	**Investigator Honest?**	**Notation**
✓	✓	✓	C U I
×	✓	✓	$\bar{C}$ U I
✓	×	✓	C $\bar{U}$ I
✓	✓	×	C U $\bar{I}$
✓	×	×	C $\bar{U}\bar{I}$
×	✓	×	$\bar{C}$ U $\bar{I}$
×	×	✓	$\bar{C}\bar{U}$ I
×	×	×	$\bar{C}\bar{U}\bar{I}$

Table 1: Combination of Collusion

CUI For the first case, where everybody is honest, there is nothing to worry about. In that case, there is no chance of evidence contamination.

$\bar{\text{C}}$UI A CSP can produce incorrect PPDP, but this is not a feasible scenario, as there is no benefit for CSP to be malicious alone.

$\mathbf{C\bar{U}I}$ In this case, where only the user is dishonest, she can delete a file from his cloud storage and later claim that she did not have that file. For example, a malicious user might have hosted a child pornography site, and later she deleted all the evidence. If the investigator comes with some contraband images, then using our scheme he will be able to verify whether the user actually possessed those files or not. The CSP picks up the last Bloom filter entry, generates the hash of the evidence that the investigator provided, and by comparing it with the retrieved Bloom filter it can say whether the user is honest or not. It proves our claim for the I1, I2, and I3 integrity properties when the user is malicious alone.

$\mathbf{CU\bar{I}}$ In this scenario, where only the investigator is dishonest, she can come up with a false evidence to frame an honest user. Our system will prevent this case from happening. Using the last Bloom filter of the user, the CSP can easily verify that the false evidence does not actually present in user's storage. A dishonest investigator can also deny a valid evidence, but using our system, the users will be able to establish their claim. Hence, we can ensure the I1, I2, and I3 integrity properties when the investigator is malicious alone.

$\mathbf{C\bar{U}\bar{I}}$ In this case, a malicious user can collude with a dishonest investigator to make herself free from an accusation. However, as long as the CSP does not collude with them, user or investigator cannot prove their false claim. If user and investigator agree on removing an evidence, they cannot prove it to court as long as the PPDP is valid. Thus, our system ensures the I1, I2, and I3 integrity properties when the investigator and the user collude.

$\mathbf{\bar{C}U\bar{I}}$ A dishonest CSP and a malicious investigator can collude to plant a false proof. In that case, the CSP was honest before colluding with the investigator, but becomes dishonest afterwards. Which means, the proof generated previously were correct but the proof generated after the colluding point will be incorrect. Here, we are proposing a scheme for proof of past data possession, that means the proof were valid at the time of generation so that nobody can establish an invalid evidence after a valid proof generation. Using the advance version of PPDP generation, we can even block the future false proof generation for this collusion combination. A CSP or an investigator cannot sign the file by user's private key. Therefore, storing the signature of the file in the Bloom filter can prevent producing any false evidence by a colluding CSP and an investigator.

$\mathbf{\bar{C}\bar{U}I}$ A dishonest CSP and a malicious user can collude together to remove valid evidence or plant invalid evidence. Before colluding with the user, the CSP generated valid proof. This proof can detect any deletion of past data. However, after colluding with the user, CSP can produce invalid proof. This invalid proof will still detect the removal of data, which was in user's storage before the colluding point but will not work for the data, which the user owned after collusion.

$\mathbf{\bar{C}\bar{U}\bar{I}}$ Finally, a malicious user can collude with a CSP and an investigator to prove her honest. Even all of the three parties collude, they cannot come up with any invalid evidence for which a valid proof has already been published by the CSP. However, they can alter the new evidence when the CSP is dishonest and publishes invalid proof of the new evidence.

In the last three cases – $\bar{C}$U$\bar{I}$, $\bar{C}\bar{U}$I, and $\bar{C}\bar{U}\bar{I}$, where the CSP is also dishonest we can ensure the I4 and I5 integrity properties by our scheme. As the published hash of bloom filter is signed with the CSP's private key, it cannot repudiate the proof (I5). Nobody, other than the CSP can use that private key to sign the hashed bloom filter. Hence, after decrypting the signed value, if it matches with the hashed bloom filter value, the CSP cannot repudiate the published value. Additionally, if the CSP comes up with a false $PPDP_u$ in place of a published PPDP$_{\mathrm{u}}$, then it will be easily detected. In that case, the $H(DS_u)$ of the published $PPDP_u$ and the $H(DS_u)$ of the false $PPDP_u$ will not be same. As the CSP has already signed the $H(DS_u)$ of the published $PPDP_u$ using its private key, it cannot deny the published value.

The published bloom filter contains the hash of all the previous evidence. If the published bloom filter contains the hash of a file, then the CSP cannot deny hosting that file (I4) when the probability of false positive rate is zero.

Besides the integrity properties, our system also ensures the two confidentiality properties – C1 and C2, which will ensure cloud user's privacy. A CSP publishes the hashed value of the bloom filter and the signature of that. As the hash function ensures the one-way property, adversaries will not be able to know about the bloom filter content from the hash value (C1). As the CSP publishes the PPDP everyday, adversaries can try to know about the evidence change-history from the regularly published proof. However, it is not possible to know about the change-history from the hashed value.

We can defend the repudiation by user attack for multi-tenancy case by using the advanced version of PPDP generation, where the user's signature of the file is stored. As the key for the signature is private to the user, nobody other than the user can sign a file using the secret key. So, if a positive match found for a signature of an evidence, the user cannot repudiate the file possession.

3 EVALUATION

To evaluate the feasibility and the performance of our scheme, we first set up a ftp server in an Amazon EC2 micro instance. Then we implemented our proof-of-concept application using JDK 1.6 and MySQL Community Server - 5.1.53. We upload files from a personal computer with Intel Core-i5-24305 CPU @ 2.40 GHz processor and 8GB RAM. We configured the Bloom filter with 0.01% false positive probability for 1000 element and used seven hash functions to maintain this configuration. We used RSA (1024 bit) for signature generation and SHA-1(160 bit) hash function for hashing. As the unique information of a user, we used the user's email address. We created a directory watcher, which watches the directories of each users. Whenever any new file is uploaded to user's directory, we hash that file along with the user information and insert into the latest bloom filter entry of that user.

Associated Overhead: For PPDP generation, we follow the option B of the process flow, i.e., we did not check the the number of elements in bloom filter before inserting the proof. To check the overhead of inserting the proof of a file in our system, we first upload 1000 files to our ftp server and measure time for each file. File size was distributed in log normal distribution. The lowest and the highest file size was 693 KB and 13843 KB respectively. Then we again upload the files to the server, but now in a directory which is being watched by our directory watcher. This time, we calculate the time of each file including the time to insert the proof of the file. From the difference of the this two time, we calculate the % overhead associated with inserting the proof of the file. We run our experiment for several times to measure the average overhead. Finally, we run the same experiment for signature-based Bloom filter. Figure [4] depicts our findings. Without signature, the overhead varies from 2.1959755% to 0.132992189% and when the file size crosses 4.5 MB, it varies from 0.2% to 0.3%, which is significantly low. For signature-based Bloom filter the overhead varies from 0.137635076% to 3.733947515% and after 6 MB, it varies between 0.3% to 0.6%. So there is a slight increase in overhead for the signature-based Bloom filter.

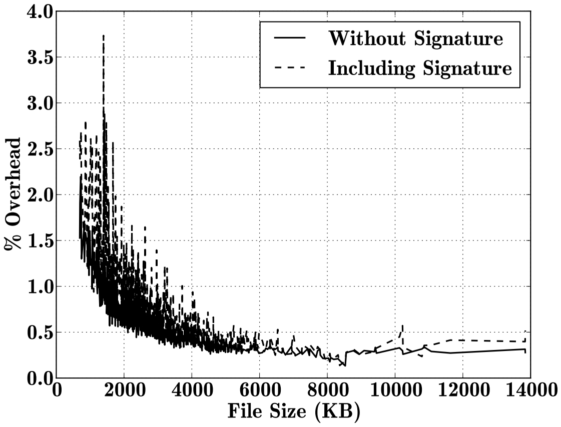

Figure 4: Overhead associated with time needed to insert the proof of data possession.

Proof Checking: To measure the performance of proof checking, we consider both the true positive matching and true negative matching. By true negative matching, we mean that the file was not present in the PPDP and our system detects it correctly. And by true positive matching, we mean that the file was actually present in the PPDP and the application detects the presence correctly.

To test the true negative matching time, we use the same file set, that we have used to measure the overhead. First, we choose such a PPDP, that the files are not present in that proof. Then we run the proof checking program with all the files and measure the time to find a true negative match. We run this test several times to get the average time. Figure [5] depicts our result. From the graph, we can observe that the time increases with the file size. The more the file size is, the more time is required.

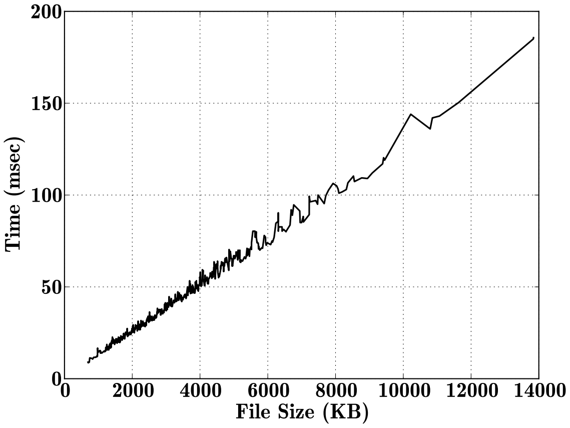

Figure 5: Average time required for true negative matching of different file size. Required time increases with the increase in file size.

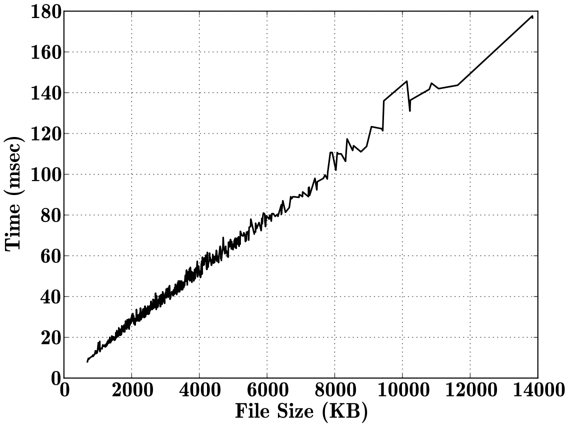

Figure 6: Average time required for true positive matching of different file size. Required time increases with the increase in file size.

Before measuring the true positive matching time, we first insert all the files in our system. That means the latest PPDP contains the proof of all the files. We select this last proof and run the proof checking program again for all the files. We find a true positive match for all of the files and note the time. Figure [6] illustrates the average time for finding a true positive match for different file size. From the graph, we can state that the finding true positive matching time also depends on the file size. For larger file size we need higher time.

In both true positive matching and true negative matching, the required time increases with the increasing file size. And the reason behind that is the hashing time. The time of hashing is proportional with the file size. In our program, to calculate the bit positions, we used 7 hash functions. Hence, it is obvious that the matching time will increase with the increase in file size.

Storage Overhead: In our experiment, we used 0.01 false positive probability for 1000 elements, which require 10099 bits, i.e., 1262.375 byte. We maintain one bloom filter for one user. Also we are publishing the Bloom filter every day, i.e., we need 1262.375 byte storage for each user everyday. For n number of users we need *n*1262.375* byte storage everyday. This is significantly low, because implementing our scheme will ensure the preservation of data possession proof without preserving the file itself. For example, if on average every user deletes 5 mega bytes file daily, then everyday our scheme will save *n*4.99* mega byte storage, but still preserves the proof of data possession.

V DISCUSSION

In this section, we discuss the applicability of PPDP scheme in cloud forensics and how it can contribute in build a regulatory compliant cloud.

1 APPLICATION IN CLOUD FORENSICS

The PPDP scheme will be applicable in cloud forensics and help the investigators to justify the evidence. In our scheme, CSPs will play a vital role. When storing a file in disk, they need to do some additional work and need to provide a new feature in management console to publish and verify the proof of past data. Storing the proof will need some extra storage for the CSP, but on the other side they can earn money from forensic agency by providing forensics-as-a-service. In this model, the CSPs do not need to store the evidence themselves. They only need to store the proof of the evidence. As a result, it will actually be quite cost effective to provide forensic-as-a-service. This is a continuous synchronization approach, but without preserving the data. If a malicious user deletes her secret data from the cloud environment, the CSPs can still preserve the proof of the record without preserving the data itself. Later, if an inspector gets the same data from any other source, he can check whether the data was owned by a particular malicious user or not. If a malicious investigator produces some false evidence, the user will also get the chance to prove it false. For cloud storage service, users will get the facility to verify the integrity of their data. If a user is suspicious about the integrity of a file, he can check the validity of the file from the management console.

Publishing the signed hash of a Bloom filter also ensures the trustworthiness of the CSP. If any user or investigator objects about the honesty of the CSP, it can defend this claim by providing the Bloom filter content to the court. If the hash of this Bloom filter content is the same as the published hash (*H(DS)*), then we can treat the CSP as honest. If the CSP is not trustworthy and had changed the Bloom filter content, then the Bloom filter content submitted to the court cannot produce the same *H(DS)*, which was signed and published previously.

2 REGULATORY COMPLIANT CLOUD

The Sarbanes-Oxley (SOX) Act 2002 mandates public companies to provide disclosure and accountability of their financial reporting, subject to independent audits [21]. While this act introduced major changes to the regulation of financial practice and corporate governance, it also brings new challenges for cloud computing. The SOX act mandates that financial information must be reside in an auditable storage system. Unfortunately, the CSPs do not provide such a service at present. Business organizations cannot move their financial information to a cloud as clouds do not comply with the act. The SOX act mandates accurate financial disclosure. For example, title VIII of the SOX act – the Corporate and Criminal Fraud Accountability Act [22] states specific criminal penalties for manipulation, destruction or alteration of financial records. But in today's cloud infrastructure, auditors cannot verify the authenticity of financial reports provided by the corporations. PPDP-enabled cloud infrastructure can provide this functionality. Implementing our scheme will help the auditors to verify whether a corporation actually possesses the records or it produces false reports. Hence, implementing the PPDP scheme will be a big step towards building a SOX-compliant cloud. Building a SOX-compliant cloud will open a new business model in cloud computing. Currently, business organizations are investing large amount of money in buying SOX-compliant storage. SOX-compliant cloud will reduce this cost substantially.

VI RELATED WORK

Cloud forensics is a relatively new topic. Several researchers have proposed solutions to overcome some of the challenges of cloud forensics. Delport et al. focused on isolating an instance to mitigate the multi-tenancy issue [23]. Isolation is necessary because it helps to protect evidence from contamination. The first technique, that is proposed is instance relocation. The second technique is server farming, which can be used to re-routing the request between user and node. The third technique is failover, where there is at least one server that is replicating another. The last approach of isolating an instance is to place it in a sandbox. To overcome the problem of volatile data, Birk and Wegener mentioned about the possibility of continuous synchronization of the volatile data with a persistent storage [6]. However, they did not provide any guideline or any practical implementation of the procedure.

Provenance provides the history of an object. By implementing secure provenance, we can get some important forensic information, e.g., who owns the data at a given time, who accesses the data and when. Secure provenance can be useful for ensuring the chain of custody in cloud forensics as it can provide the chronological access history of evidence, how it was analysed, and preserved. There have been some works for secure provenance in cloud computing [24, 25], but no CSP has practically implemented this mechanism yet.

As traditional technique of signature detection for digital forensic does not apply in the distributed architecture of cloud computing, Hegarty et al. proposed a distributed signature detection framework that will facilitate the forensic investigation in cloud environment [26]. Current model of file storage comprised of two components : Meta data Servers (MDS) and Object Storage Devices (OSD). The hash value of each file is stored in the MDS as an e-tag and integrity is checked each time after uploading / downloading a file. In the proposed framework, the first step is to send a list of target buckets to the Forensic Cluster Controller (FCC) along with a file containing the target MD5 hash values. The FCC then initializes and queries to Analysis Nodes (AN) for getting the number of files contained in targeted bucket. Upon receiving the round one signature file from FCC, each AN retrieves the e-tags of the bucket. The signatures in the round one signature file are compared with signatures, generated from the e-tags by the AN. After getting feedback from all ANs, FCC terminates the ANs. The authors tested their framework by two ways – using Amazon S3 and by emulating a cloud platform locally.

There are other important challenges orthogonal to our work. To overcome the data acquisition problem, researchers have proposed creating infrastructure support in the cloud management plane and provide an Application Programming Interface (API) (offered by the CSP) to give investigators access to various logs. Dykstra and Sherman recommended a cloud management plane for using in the infrastructure as a service (IaaS) model [27]. From the console panel, customers, as well as investigators can collect VM image, network, process, database logs, and other digital evidence, which cannot be collected in other ways. The problem with this solution is that, it requires an extra level of trust – investigators must trust the management plane. In traditional evidence collection procedure, where we have access to physical system, this level of trust is not required. To get necessary logs from all the three cloud service models (infrastructure/platform/software as a service), Bark and Wegener proposed that the CSP can provide network, process, and access logs through a read-only API [6]. By using these APIs, investigators can acquire valuable informations.

Virtual Machine Introspection (VMI) can also be helpful in forensic investigation. VMI is the process of externally monitoring the runtime state of VM from either the Virtual Machine Monitor (VMM), or from some virtual machine other than the one being examined. By runtime state, we are referring to processor registers, memory, disk, network, other hardware-level events. Through this process, we can execute a live forensic analysis of the system, while keeping the target system unchanged [28]. In this work, Hay and Nance showed that if a VM instance is compromised by installing some rootkit to hide the malicious events, it is still possible to identify those malicious events by performing VMI.

Though the above solutions are important for cloud forensics, none of the existing research efforts in cloud forensics address the problem of proving past data possession. In this paper, we took the first step towards providing a solution for the past data possession problem. Combining all the previous solutions and our scheme will drive towards building a forensic-enabled cloud.

VII CONCLUSION AND FUTURE WORK

With the increasing use of cloud computing, cloud forensics has attracted the attention of the security and forensics research community. Researchers have been exploring the challenges and proposed some solutions to mitigate the challenges. Several researchers proposed continuous synchronization of cloud data to overcome the forensic investigation challenges, though no scheme has been proposed yet about continuous synchronization. Proof of Past Data Possession can be the solution of continuous synchronization. It will allow users and investigators to verify the possession of incriminating evidence by a suspect at a past time. In this paper, we proposed a scheme of PPDP, described how it can be utilized in cloud forensics, and evaluated the performance of our scheme through a proof-of-concept application. Our experimental result indicates that the overhead associated with generating the proof of evidence is very low and practically implementable by the CSP.

In future, we will implement our scheme on our own cloud testbed using the Eucalyptus [29] framework. By implementing our own cloud testbed, we can perform more sophisticated test and will be able to make a practically usable application. After our successful implementation, we will collaborate with a CSP to deploy the scheme in a real-life cloud infrastructure.

ACKNOWLEDGMENTS

This research was supported by a Google Faculty Research Award, the Office of Naval Research Grant #N000141210217, the Department of Homeland Security Grant #FA8750-12-2-0254, and by the National Science Foundation under Grant #0937060 to the Computing Research Association for the CIFellows Project.

References

[1] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud migration: A case study of migrating an enterprise it system to iaas", in *proceedings of the 3rd International Conference on Cloud Computing (CLOUD)*. IEEE, 2010, pp. 450–457.

[2] Market Research Media, "Global cloud computing market forecast 2015-2020", `http://bit.ly/SS3Jf0`, [Accessed July 5th, 2012].

[3] Gartner, "Worldwide cloud services market to surpass $68 billion in 2010", `http://bit.ly/Rqh4L2`, 2010, [Accessed July 5th, 2012].

[4] INPUT, "Evolution of the cloud: The future of cloud computing in government", `http://bit.ly/ZgpTKz`, 2009, [Accessed July 5th, 2012].

[5] FBI, “Annualreport for fiscal year 2007”, 2008 Regional Computer Forensics Laboratory Program, 2008, [Accessed July 5th, 2012].

[6] D. Birk and C. Wegener, “Technical issues of forensic investigatinos in cloud computing environments”, *Systematic Approaches to Digital Forensic Engineering*, 2011.

[7] “Insider trading”, http://1.usa.gov/UsdOmI, [Accessed August 12th, 2012].

[8] J. Wiles, K. Cardwell, and A. Reyes, *The best damn cybercrime and digital forensics book period*, Syngress Media Inc, 2007.

[9] K. Kent, S. Chevalier, T. Grance, and H. Dang, “Guide to integrating forensic techniques into incident response”, *NIST Special Publication*, pp. 800–86, 2006.

[10] D.A. Lunn, “Computer forensics–an overview”, *SANS Institute*, vol. 2002, 2000.

[11] J. Robbins, “An explanation of computer forensics”, *National Forensics Center*, vol. 774, pp. 10–143, 2008.

[12] K. Ruan, J. Carthy, T. Kechadi, and M. Crosbie, “Cloud forensics: An overview”, in *proceedings of the 7th IFIP International Conference on Digital Forensics*, 2011.

[13] Clavister, “Security in the cloud”, http://bit.ly/UeAzo3, [Accessed July 5th, 2012].

[14] Amazon, “Zeus botnet controller”, http://amzn.to/TQOYe4, [Accessed July 5th, 2012].

[15] J.R. Vacca, *Computer forensics: computer crime scene investigation*, vol. 1, Delmar Thomson Learning, 2005.

[16] M. Taylor, J. Haggerty, D. Gresty, and R. Hegarty, “Digital evidence in cloud computing systems”, *Computer Law & Security Review*, vol. 26, no. 3, pp. 304–308, 2010.

[17] G. Grispos, T. Storer, and W.B. Glisson, “Calm before the storm: The challenges of cloud computing in digital forensics”, *International Journal of Digital Crime and Forensics (IJDCF)*, 2012.

[18] D. Reilly, C. Wren, and T. Berry, “Cloud computing: Pros and cons for computer forensic investigations”, 2011.

[19] B.H. Bloom, “Space/time trade-offs in hash coding with allowable errors”, *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[20] “Bloom filters the math”, http://bit.ly/SWOyEI, [Accessed November 4th, 2012].

[21] “The sarbanes-oxley act”, http://www.soxlaw.com/, [Accessed August 12th, 2012].

[22] “Corporate and criminal fraud accountability act”, http://1.usa.gov/Q2RkYe, [Accessed August 12th, 2012].

[23] Michael KÂ¨ohn Waldo Delport, Martin S. Olivier, “Isolating a cloud instance for a digital forensic investigation”, in *Information and Computer Security Architecture (ICSA)*, 2011.

[24] K.K. Muniswamy-Reddy, P. Macko, and M. Seltzer, “Provenance for the cloud”, in *proceedings of the 8th USENIX conference on File and storage technologies*. USENIX Association, 2010, pp. 15–14.

[25] K.K. Muniswamy-Reddy and M. Seltzer, “Provenance as first class cloud data”, *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, pp. 11–16, 2010.

[26] R. Hegarty, M. Merabti, Q. Shi, and B. Askwith, “Forensic analysis of distributed data in a service oriented computing platform”, in *PG Net The 10th Annual Postgraduate Symposium on The Convergence of Telecommunications, Networking & Broadcasting, Liverpool John Moores University*, 2009.

[27] J. Dykstra and A.T. Sherman, “Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques”, *DoD Cyber Crime Conference*, January 2012.

[28] B. Hay and K. Nance, “Forensics examination of volatile system data using virtual introspection”, *ACM SIGOPS Operating Systems Review*, vol. 42, no. 3, pp. 74–82, 2008.

[29] Eucalyptus., “Cloud computing framework”, http://www.eucalyptus.com/, [Accessed July 5th, 2012].