SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2016; **9**:571–582
Published online 19 June 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1056

SPECIAL ISSUE PAPER

Time synchronization: pivotal element in cloud forensics

Nikolaos Marangos, Panagiotis Rizomiliotis* and Lilian Mitrou

Department of Information and Communication Systems Engineering, University of the Aegean, Karlovassi GR-83200, Samos, Greece

ABSTRACT

Cloud computing (CC) is the new trend in computing and resource management. This architectural shift toward thin clients and the centralized on-demand provision of computing resources aspires to offer significant economical benefits to its users. However, the adaption of the CC model has forced many times the IT industry and the academia to revisit most of the traditional tools and technologies. The last few years, it has been identified that one of the computer branches that has been most affected by the CC model is Digital Forensics, one of the main law enforcement tools in the cyberspace. In this context, a new security area was born, the so-called cloud forensics (CF). In this paper, we investigate the impact that the CC model has on the trustworthiness of one of the main CF sources of information, the log-files. More precisely, we bring forth a crucial but rather underestimated problem, the problem of accurate log-records timestamping. The synchronization of time (stamps) is of major importance for the investigation logs to be used as source of evidence. We show that this requirement is not easy in the cloud context. We demonstrate that the main features of CC render existing time synchronization techniques inadequate, and we provide a list of guidelines toward a CF aware timekeeping system. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS

cloud computing; digital forensics; timekeeping

***Correspondence**

Panagiotis Rizomiliotis, Department of Information and Communication Systems Engineering, University of the Aegean, Karlovassi GR-83200, Samos, Greece.

E-mail: prizomil@aegean.gr

1. INTRODUCTION

Cybercrime, a severe threat which increases rapidly, is leading to substantial financial losses and causing major damage to the world economy. Referring to Norton's 2012 report [1], the scale of consumer cybercrime victims has increased to 556 million per year, or in other words, we have more than 1.5 million of victims per day. This elation of crime has significantly affected user's trust toward electronic services. The European Commission has recently reported that "around 38% of European internet users have changed their behavior because of cyber-security concerns: 18% are less likely to buy goods online, and 15% are less likely to use online banking according to a recent survey" [2].

Cybercrime is an international lash, because online criminals can mount an attack practically from anywhere. Thus, to efficiently protect the citizens and to apply cyber-space laws, a common international approach and strategy is required. One of the main pillars of such an effort is the area of IT security, called digital forensics (DF).

Digital forensics is the branch of computer science that focuses on developing evidence pertained in the digital world for use in civil or criminal court proceedings [3]. It can be seen as the natural descendant of computer forensics that is targeting only computer systems for acquiring potential evidence. However, the enormous increase of digital devices population during the last two decades rendered computer forensics insufficient. In DF, practically every electronic device storing, processing, transmitting, and receiving digital data is searched and analyzed. Nowadays, DF investigators have a variety of tools at their disposal, proprietary and open source, to conduct their analysis. This analysis is organized based on one of the many DF methodologies that exist [4].

Cloud computing (CC) is the new trend in computing and resource management, an architectural shift toward thin clients and conveniently centralized provision of computing resources. CC can be very cost-effective, as an organization can have remote access in a pay-per-use manner to exactly the resources that it needs and only for the necessary period of time. In such a business model, large-scale cost reduction can be achieved as, instead of building and maintaining proprietary IT infrastructure, this responsibility is delegated to a third party, called the cloud service provider (CSP). However, migrating to cloud and

taking advantage of the economic benefits that CC can offer does not come at no cost. As the traditional security mechanisms have been found insufficient, the CC clients are invited to accept new security risks. In this line, DF has been identified as one of the main security related areas that the use of CC has negative repercussions [5–7].

Cloud computing is a constantly evolving technology, and its main features differentiate it significantly from the traditional IT systems rendering many of the contemporary DF tools and methodologies insufficient [5]. In order to cope with this deficiency, a new branch of DF was recently born, the so-called cloud forensics (CF) [7,8]. The last few years, a significant research activity has taken place aiming to investigate the applicability of the existing DF methodologies and tools in the CC paradigm, their adaptation to CC, if possible, or the introduction of new techniques if necessary.

Traditional DF methodologies share a common general structure, and they are divided in three main phases: the preparation phase, the investigation phase, and the presentation phase [4]. Despite their differences, all these methodologies have practically a common research starting point: log-files. Log-files are to DF investigators what fingerprints are to traditional crime scene investigators or what financial ledgers are to auditors [9]. However, this crucial primitive source of information, in order to be useful and admissible as evidence, has to satisfy some strict requirements.

In this paper, we investigate the influence that CC has on the trustworthiness of log-files. We analyze the challenges that a logging system designer faces, and we focus on the importance of the right timing in the log-records. Time is the main metadata that completes a log-record. However, the difficult problem of timekeeping in traditional systems becomes an extra headache for CC logging system designers, and unfortunately, the issue of time synchronization has been underestimated from the CF research community. Furthermore, because any typical CF investigation leads to the court of law, we have also written the current status of the related legislation, having in mind that the paper aims to address technical audience.

1.1. Our contribution

- We identify the main technical challenges that a CF logging system designer faces and the security requirements that such a system must satisfy.
- We bring forth the problem of timekeeping in CC and its impact to CF investigation.
- We present the existing time synchronization techniques for CC, and we evaluate their adequacy for log-files timestamping in the context of CF.
- We provide a list of guidelines toward the design of CF aware timekeeping techniques for CC.

1.2. Paper organization

This paper is organized as follows. In Section 2, we introduce CC and its main features. In the next section, we list the technical challenges that a CF logging system designer faces, and we highlight the security requirements that this system must satisfy. In Section 4, the difficulties in timekeeping that appear in CC are analyzed. We examine these difficulties from the perspective of both the CSP and client, focusing mainly on the IaaS model. In Section 5, we point out the criticality of time accuracy in a typical CF investigation. Also, we report a real criminal case as an example. In Section 6, we report the timekeeping techniques that are mainly used in CC. In Section 7, we examine and evaluate the existing techniques in terms of their reliability and accuracy. We discuss our outcomes, and based on our analysis, we propose a list of recommendations and guidelines for the design of a CF aware timekeeping system. Finally, in the last section, directions for further research are proposed.

2. BACKGROUND

In this section, we briefly introduce the CC model focusing on the presentation of its features that mainly affect CF.

So far, three main service delivery models have been defined: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). IaaS is the most characteristic of the three. In IaaS, CSP offers resources in the form of virtual machines (VM), controlled by the special software called the hypervisor or else the virtual machine monitor. The hypervisor is administered by the CSP, and the client rents access to a VM or a virtual resource [10]. On the other hand, the PaaS model aims to provide low cost development and deployment platforms. A client rents from a CSP software building blocks and development tools in order to create new applications. Finally, in the SaaS model: "software is deployed as a hosted service and accessed over the Internet" [10]. That means that, a user rents from a CSP access to an application that the CSP is responsible for its management and proper execution.

In CC, there are four deployment models [10], namely the private, the public, the community and the hybrid cloud model. A private cloud is operated solely by an organization or by a business, while a public cloud is owned by a CSP, and it is available to anyone who wants to buy its services. In the community model, a group of organizations or businesses, which have shared concerns and interests, operate the cloud and, finally, in the hybrid approach, some of the previous deployment models are combined.

Cloud computing is based on a plethora of existing technologies, like virtualization, grid computing, service-oriented architectures, distributed computing, and broadband networks. This combination aims to offer resources as a service and on-demand from a pool of resources using a broad network access. The main features of CC can be summarized as follows.

Multi-tenancy: The main feature of the CC paradigm is multi-tenancy, that is, different clients are sharing the same physical resources.

Several responsibility levels: Depending on the service delivery model, the user and the CSP have different control

Security Comm. Networks 2016; **9**:571–582 © 2014 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

over resources and, correspondingly, different level of responsibility. In our case, logging responsibility is determined by the model.

Heterogeneity: CSP use various hardware and software resources to build cloud environments. These resources may differ between CSPs causing sever interoperability problems. For instance, a client might subscribe to an IaaS from one CSP, combine it with a PaaS from another CSP, and demand pieces of SaaS from a third cloud vendor. This heterogeneity influences the type of log-records that are stored, and usually, resource virtualization cannot provide sufficient high-level system homogeneity.

Virtualization: The clients have the illusion of using a dedicated machine. However, there is significant difference in efficiency between physical and virtual resources.

Mobility and distributed resources: CC aims to be a global phenomenon by potentially harvesting widely dispersed infrastructural resources. That is that the physical hardware that hosts a cloud service, in any of the three service delivery models, can be located anywhere around the globe, independently of the provider's legal address and, the same CSP can have several physical machines that each one of them is located in a different place. To give a more holistic image of the CC's particularity, a client can rent resources from several such physical machines at same time or he can mitigate his virtual machine to different physical locations during the same day.

3. LOGGING FOR CLOUD FORENSICS

One of the major operations in every application, and in any IT infrastructure in general, is producing log-records. Servers, operating systems (OSs), routers, database management systems, and network protocols all include procedures for effective events recording like malfunctions, user log in–log out, and so on. This information is stored in a log-file, that is, a file that lists actions that have occurred in an IT infrastructure. Log-files were first created by application developers, including the OS, in order to effectively debug the operations of an application. Over the years, log-files became more and more useful. Log-files are stored for specific business reasons or to satisfy legal requirements. They help system administrators to discover possible problems to their infrastructure; they report the reason for an unexpected termination of an application and so on.

In DF, log-files are one of the primary sources for evidence retrieving from a suspect device [9,11–13]. In the vast majority of the cases, the log-files constitute the starting point of an investigation. In a typical DF investigation, there are two different strategies: top-down and bottom-up [14]. In the top-down approach, the investigator starts after the occurrence of an attack, and he investigates the log-files in order to discover potential evidence. In the bottom-up approach, log-files are examined, usually by automated tools, with the view to find potential threats or policy violation. In any of the two strategies, the DF investigator collects from logs-files the events that have occurred and creates a time line containing all the suspicious actions to fully describe the incident. After that, one of the well-established DF process is followed.

The efficiency and the reliability of a DF investigation rely heavily on the completeness of the log-files, as well as on the protection of their integrity and confidentiality. However, the design and implementation of a reliable logging system is a difficult task for traditional systems that becomes a real challenge for CC, given its dynamic and multi-tenant nature. A criminal, user of the cloud, after committing an illegal action, can simply delete his VM and with it all potential evidence [15]. Thus, because of the volatile nature of virtual resources, log-files are available only for a certain period. This is just a small, but characteristic, example of the difficulties that a CF investigator faces.

The use of many different CSPs may cause interoperability problems, which directly affect logging. For instance, a client might subscribe to an IaaS from one CSP, combine it with a PaaS from another CSP, and demand pieces of SaaS from a third cloud vendor. Also, the variety of controls in the three deployment models has also an impact on a logging system. This occurs because each one of the models grants different access rights to the underlying infrastructure and thus to its log-files. For example, a SaaS user is able to access only application log-files and only if this is allowable from his CSP.

Most CSPs offer applications for collecting and displaying the contents of the log-files. For example, Amazon CloudFront provides the LogAnalyzer [16]. By using this tool, the user can generate reports containing information regarding object popularity, traffic volume, and IPs. However, although general logging rules are followed, it seems that these applications are insufficient for a CF investigation [17–19]. In the rest of this section, we describe the main technical challenges that a logging system designer faces and the security requirements that such a system must fulfill.

3.1. Technical challenges

The circle of life of a log-file is divided in two phases: the log-record generation and the log-record management. In the first phase, the designer has to decide on the exact fields of the log-records, that is, which type of events must be logged and which type of information related to a specific event must be stored. Then, the collected information must be completed with crucial metadata for a CF investigation like the exact time that the event occurred. The designer has to decide also, on the location that a log-file is maintained and a choice between a centralized and distributed storage must be made. In the second phase, the management of the log-record is performed. The retention period of a record is decided, as well as the access policy is defined, that is, who has the right to read a record, which fields exactly and what processing rights he has. In more details, the technical challenges that a logging system designer faces are as follows.

3.1.1. Log-record's contents.

There is no standard description of the fields that a log-record must possess and usually, the contents of log-files differ depending on the application programmer. As a rule of thumb, each record should contain "who," "what," "when," "why," and "how" something happened [9]. Because of the mobility that characterizes the CC paradigm, "where" must be answered as well, as a user's VM can migrate from a physical machine to another over time. In general, a log-record should encompass as many identifiers as possible [20], and these identifiers must uniquely identify the person, and his location, which carried out the recorded operation. The outcome must be solid and indisputable. For instance, in the case of Amazon CloudFront, the fields of x-edge-location, c-ip, cs (Host), cs (Referrer), cs (User Agent), and x-cf-client-id are potential sources of acquiring information about the location, the process, and the person who carried out something. Unfortunately, the aforementioned fields are not sufficient to fully cover the identifiers that a log-record should have for a CF investigation [15,21].

3.1.2. Log-record's source.

Referring to the source of logging, there are two types of logs: operating system logs and network logs [14]. OS logs are also divided in two categories: system-level logs and application-level logs [22]. System-level log-files are generated by the OS itself and usually contain information like users' log in and log out, device changes, device drivers, system changes, system calls, and so on. On the other hand, application-level log-files are generated by the applications hosted by the OS. The contents of such a file depend on the respective application developer. Finally, network logs are created by routers, network protocols, and firewalls. They usually contain information regarding the network/Internet behavior of a specific device or network.

In CC, we have user's OS logs that are produced by the guest OS, and the applications running inside a VM and user's network logs that are recording events related to virtual devices, like virtual routers. On the other hand, there are the CSP's log-files that are produced by the hypervisors that manage a physical machine and the physical network devices, as well as, logs that are created by the CSP's management software.

3.1.3. Timestamping.

A timestamp is the information needed to identify when a certain event occurred. Each record must have such a stamp, that is, a time field, in order to report the specific time the record was created [15,23]. Unfortunately, time accuracy is not trivial to achieve in IT for a variety of reasons: servers have wrong time settings, clocks are off beam or are drifted between few seconds to many hours, and so on. In the rest of the paper, we will elaborate on the critical role that time accuracy plays in a CF investigation, and we evaluate the timekeeping state of art techniques in CC.

3.1.4. Distributed versus centralized log-file storage.

A logging system can be distributed or centralized in terms of log-file storage. In the distributed model, the log-files are maintained on the physical location that they are created. In the centralized approach, all log-records all stored together. Several, mainly file-centric, logging systems have been proposed [3,17,24–26].

3.1.5. Log-records retention period.

Usually, the data retention period is defined by data retention rules [14]. However, in CC, a significant legislation gap has been identified [15]. As a general rule of thumb, log-files are destroyed after their business use is completed.

In practice, log-files are either sequential or circular [27]. In sequential (or linear) logging, every event is written down at the beginning or at the end of the log-file. This type of logging provides information from the first day of operation of the IT infrastructure. Unfortunately, the size of these log-files is extremely big, and in most cases, the oldest logs are useless. On the other hand, in circular logging, the log-file has a predefined specific maximum size. When it reaches its maximum size, it writes down the newest entries above the oldest ones. This type of logging provides relatively small size log-files, at the expense of losing old events. Of course, deleting log-records in circular logging is subject to data retention legal restrictions.

3.1.6. Access policy.

Grading access to log-files is a very delicate issue. Log-files may contain confidential business information or privacy sensitive information. Usually, an access policy is defined, mainly dictated by the legislation and the organization's business policy. Then, subject to the privacy requirements of this policy, the log-files, partly or as a whole, are considered to be confidential. Depending on the OS, particular tools are available for accessing and viewing log-files. For example, Microsoft Windows OSes incorporate the Event Viewer, while for UNIX/Linux based OSes, there is no default log-file viewer. All these tools must have only "read-only" rights to the log-file's contents.

3.2. Security requirements

A logging system must satisfy at least three security requirements. First, the authenticity of the produced logging data must be guaranteed, that is, the source of the log-record must be legitimate, and the record must be stored unaltered, including its metadata information, like timestamps. Second, the integrity for the log-file contents must be protected, and access must be allowed by strictly applying the defined policy. In more details, the basic security requirements that a logging system must fulfill are as follows.

3.2.1. Log-record authenticity.

A log-record must be authentic in order to be admissible in a court of law [28]. That means that, the source of the record must be legitimate, and the information that contains must be unaltered during its transmission from

Security Comm. Networks 2016; **9**:571–582 © 2014 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

the source to the log-file. In order to ensure the source authenticity, there should be an indisputable way to associate the log-record with the log-record creator. By doing so, we want to avoid forged logs created by an adversary that can mislead a CF investigator. Finally, the integrity of the log-record's contents must be protected, to avoid unauthorized modifications. Unfortunately, in many cases, log sources are not properly secured, including insecure transport mechanisms, and they are susceptible to log configuration changes and log alterations [29].

3.2.2. Log-record integrity.

Protection of log-file's integrity is of paramount importance, and it has been extensively studied in the literature [9,11,14,27,30]. Any indication that implies that the file has been tampered and modified can render its content untrustworthy and useless for CF. The same fate will have any investigation outcome that it is based on the contaminated log-file. Unfortunately, in most cases, log-files are not sufficiently protected, and there is no guarantee for their reliability.

3.2.3. Log-record privacy.

The log management system must enforce the defined access policy. This requirement is not directly related to CF, as the violation of log-file's confidentiality cannot put at risk its role in a forensic investigation. However, accessing a file record must be logged as well, as it can be proved valuable information.

4. TIMEKEEPING PROBLEMS IN CLOUD COMPUTING

Timekeeping is a challenging problem for contemporary IT systems that becomes a real nightmare for virtualized environments. There are two different perspectives: the CSP 's and the client's. In the first case, we are investigating the problem of correct timestamping for CSP's logs and, in the second case, for all the other log-files, that is, the client's log-records. The two cases have significant differences, but their impact on timekeeping is accumulative.

4.1. Cloud service provider timekeeping

A CSP has two aims: he wants all his servers to be synchronized, that is, all the physical machines must have the same time, and all his servers' clock to have the correct time. This timekeeping procedure faces all the classic difficulties of a distributed system. The main challenge is related to the distance between the physical machines. It is very likely that the servers are located even in different continents and that the network latency may influence some of the contemporary synchronization technologies. We discuss, in more detail, the cloud synchronization problem and the techniques that are used today in Section 6.

4.2. Virtual machine timekeeping

We will focus mainly on the IaaS model. The analysis of the two other service delivery models is similar.

The guest OS faces the same challenges in timekeeping when running either in a virtual or in a physical machine: it has to initialize the wall-clock to the correct time and, then, to keep it always updated. Timekeeping has been extensively studied for physical machines, and the currently used techniques are well performing. However, in the virtual world, there is a variety of reasons that makes keeping time accurate inside a VM a much more tricky procedure. Most of the difficulties originate from the special features of CC. In Section 6, we report the techniques that are used for timekeeping inside a VM, and we evaluate them from the scope of a CF investigator. More precisely, the main characteristic of CC influence timekeeping as follows.

Multi-tenancy: Traditionally, operating systems measure the time passing in two ways: tick counting and without tick counting. Tick counting approach is dominant. The guest OS uses a hardware clock only for initializing the wall-clock time when booting, and then it keeps time by using special routines based on interrupts counting. Also, the guest OS assumes the existence of a hardware device that sends, with a known rate, interrupt signals. Counting these interrupts, the OS can compute the elapsed time.

Usually, the OS does not have consciousness that it runs in a VM and that it actually shares the hardware with other co-hosted VMs. The hypervisor is designed to hide this fact and create the illusion of sole hardware use. However, in practice, this multi-tenancy influences the performance of the routines that the guest OS uses for time counting, leading to time drifts. For instance, at the moment that it must generate an interrupt, it may not actually be running, resulting to an accumulation of backlog of timer interrupts. Thus, OS time falls behind real time whenever there is a timer interrupt backlog. Clearly, there is a scalability issue as more and more VMs are running on the same physical machine.

Virtualization: Usually VMs are saved and suspended. When a VM is resumed from suspension or restored from a snapshot, it continues running as it was before. That means that, the memory and the run state are put as they were before. The guest OS is not aware that something has changed, and thus, the clock remains at the value that it had at the moment of suspension or snapshot, and it continues running with the internally calculated time.

Mobility and distributed resources: Another problem derives from the difference between the time zone that the VM is running and the time zone that the hypervisor is using. Thus, when the VM, via a system call, is using the host clock for time synchronization, it does not take into consideration this inconsistency, leading to clock drifting. The problem propagates and accumulates, as the VMs migrate from one physical machine to another.

Variety of control and responsibility levels: The guest OS can perform all the operations that it usually performs

when running on a physical machine to maintain the time accurate. The most effective technique is to use a network clock synchronization protocol like the Network Time Protocol (NTP) or the W32TIME. However, it can be shown to be insufficient. For instance, usually the guest OS picks a time synchronization source that it is different from the one used by the CSP. Thus, when the Cloud controlled hypervisor tries to enforce timekeeping service to the VMs, this differentiation can confuse the guest OS. Finally, sometimes when a VM boots, it takes time for the hypervisor to initiate its virtual-RTC. By doing so, it totally depends on the CSP's for reliable timekeeping, that is, in case the physical machine's clock has wrong time, the VM will boot with the wrong time as well.

5. TIME SYNCHRONIZATION: A CLOUD FORENSICS CRITICAL REQUIREMENT

Every forensic investigation has to be carried out (also) in compliance with substantial and procedural legal requirements. Digital evidence must be relevant, authentic, reliable, and admissible based on both the integrity of the evidence itself and the integrity of the (forensic) investigation process [31]. Prosecution and judicial judgment are based upon evidence that a specific person at a specific place and time committed an unlawful act [32].

In order to be an admissible source of evidence for legal disputes, the reconstruction of event chains requires information about the act(ion) to be investigated (subjects and objects) and when both the action and its investigation took or take place. Acquiring and preserving integral evidence in cloud environment present(s) additional (technical and legal) challenges not only over its conventional counterpart but also in relation to digital forensics.

These challenges are born out of the specific characteristics of CC and mainly the ephemeral nature of the cloud as well as to the fact that data are stored "elsewhere" (in another country or in more countries) [33]. It is clear that metadata, such as time of file creation, or modification and access times, provide a valuable source of potential evidence to the forensic investigator. However, because of the nature of CC, it is uncertain where data is/was geographically located at any particular time. A further uncertainty concerns the manipulable nature of time in cloud environments: indeed, an investigator cannot "be sure evidence is not in the process of being altered in the cloud at that moment in time" as Grispos mentions [34].

In cloud environments the spatial dimension is strictly interrelated with the temporal dimension. The difficulty to access data and piece together a sequence of events encompasses the difficulty to (re-)construct an accurate timeline of events on the device, system, or network as the correct time and time zone need to be established. Time definition and time synchronization (synchronization of timestamps) have been identified [35] as an important challenge both for the detection and the preservation of evidence. The proper and admissible preservation of digital evidence requires the "reconciliation" of timing information from time stamps, for example, in metadata or application log-files of more devices or systems [34].

Defining the exact time of the act(s) to be investigated and of the respective acts is of crucial importance for the integrity of every investigation procedure. In relation to the famous "Carnivore case," the final report of the independent review pointed out deficiencies of the Carnivore System that referred not only to privacy concerns but also to its reliability: the report underlined as an important potential reliability issue that could be challenged in a (criminal) trial the fact that time stamps for the data were based on the collection computer's clock, and there was no time synchronization between the collection and control computers [36]. The correct time factor is decisive (even) for the characterization of an act as criminal, punishable or investigable, the definition of both jurisdiction and competent authorities and the compliance of their action with procedural principles and provisions.

The synchronization of time (stamps) is of major importance also for the investigation/audit logs to be used as source of evidence. A requirement that obviously is not easy to fulfill, especially in a cloud context where the cloud server and the cloud client are usually located in different time zones: not only physical locations could be in multiple geographical regions and consequently different time zones but also virtualized services may be oriented to the (respective and ephemeral/changing) time zones of their web clients. An investigation that does not take this "reality" into consideration risks not only contradicting results [34] but also the integrity, reliability, and admissibility of evidence in court.

Cloud computing poses a number of significant complications for cybercrime investigations. The existence of transnational and multinational clouds, as they are usually built, coupled with the unknown location of data raise(s) major legal concerns, which refer mainly to jurisdiction [37]. While cloud seems to be borderless per definition, both the investigation and punishment of crime and the lawsuits are strictly related to sovereignty and territorially defined jurisdiction of investigators and courts. Investigation, evidence collection, and court procedure are principally regulated by national law, which differ from country to country. Jurisdictional differences in procedural and substantive law complicate and may also hinder law enforcement, as legal enforcement authorities are dependent on cloud service providers to access the relevant data.

On the same time, even if legislators in many countries have passed laws on cybercrime, cloud-specific legislative provisions have not (yet) been adopted, and many current law procedures have not (yet) been adapted to deal with investigation "in the cloud" [38]. The intrinsically global nature of cloud storage and processing and the respective jurisdictional complications require increased multinatinal

and international cooperation and greater harmonization of substantive and procedural laws. As a legal source may serve the Council of Europe Convention on Cybercrime (2001) that has been signed by more than 50 states, including non-European states such as the USA, and ratified by the majority of them. This binding legal text has a quite broad scope providing also for procedural instruments such as search and seizure of stored computer data, interception of content data as well as trans-border access to stored computer data (with consent or where publicly available) or mutual assistance in the real-time collection of traffic data. Such provisions facilitate cooperation to investigate and prosecute crime on a multi-jurisdictional basis [23]. Despite the fact that both the definitions and the provisions of the so-called Budapest Convention reflect the state of technology 15 years ago, they may be regarded as applicable to investigations in the cloud.

5.1. A time drifting sensitive cloud forensics paradigm

In order to make clear the importance of time in a CF investigation, we shall write down an incident occurred in a typical computer forensics investigation almost a decade ago [39]. This incident can be visualized in the CC context with limited modifications.

The trigger for the investigation was an email of an individual suspected of involvement in communication of child abuse images. As usual, a warrant was obtained, the suspect was arrested, and his digital equipment was seized. A relatively simple forensic examination would take place in order to identify traces of relevant email or child abuse images in suspect's computer. Seizing in the CC paradigm can be realized by copying snapshots of the suspect's VMs.

During the police interview, the suspect failed to provide any explanation for the unlawful images. He only claimed that he was not guilty. After that he was brought to the local court.

Meanwhile, the police officers were expecting the forensic investigator to comment on the digital evidence he found in suspect's computer. When the report was presented to the prosecution, they were shocked to find out that it contained serious allegations of malpractice by the police [6]. The beginning of the report stated that "the defendant's computer [ID number] was used to access the internet after it was seized and was in police custody. Approximately 750 records on Internet access are time stamped during the six hours or so after the computer was seized {…} pages accessed included Hotmail login pages and possible child pornography site. Floppy diskettes were also used". The report stated that computer's contents were altered when it was seized by the police, indicating that, for an unknown reason, the police had planted fake evidence in suspect's computer. It concluded by stating "However I am sure that there are so many grave problems with this evidence and with all the computer evidence submitted by the prosecution, that the Court cannot safely rely on it."

Because of the severity of the report's results, a second investigation took place. For the police officers, it was sure that none of them had altered or accessed the suspect's computer. The investigator should identify the reasons for this erroneous report. After a detailed examination of the seized computer, the investigator finally found out what had really happened. The only reason for this huge problem was time. Suspect's computer clock was wrong, a fact that the first investigator missed to notice.

In CC, the suspect can possess more than one VMs. The CF investigator must extract examine all of them and extract evidence. Any difference in any of these VMs clock confuses the investigator when he tries to correlate events. Even, 10 or 20 s deviation can be fatal considering that time in the CC can drift backward or forward by default, without a standard pattern.

6. EXISTING TIMEKEEPING TECHNIQUES FOR CLOUD COMPUTING

There is a significant variety of techniques that offer time synchronization in a CC environment. In the following paragraphs, we expound the main of these techniques in more detail.

6.1. Synchronization with an external time source

Cloud service providers are using traditional solutions in order to keep their physical machines synchronized and in order for their clocks to contain the correct time (Figure 1). More precisely, network time servers are used

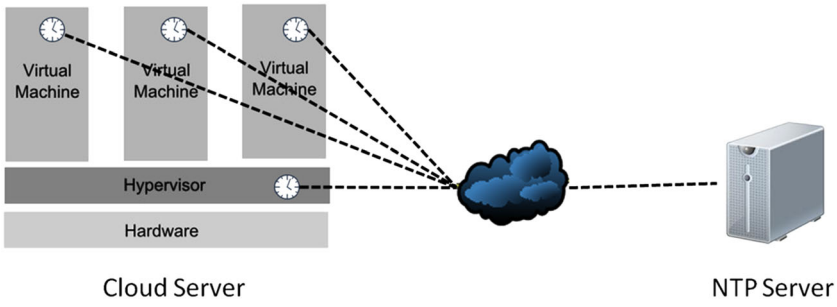

Figure 1. Synchronization with an external time source.

to collect the correct time via widely acceptable protocols, like the NTP and its Microsoft version W32TIME.

NTP: The NTP is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks [40]. NTP was designed by D. Mills of the University of Delaware. Its purpose is to provide UTC (Coordinated Universal Time) only. This means that NTP does not provide any information about time zones or daylight saving time. It offers time maintenance within tens of milliseconds over the public internet, and it is capable of achieving one millisecond of accuracy in LANs. Its current version is 4.0, and its full specifications are in [40].

Network Time Protocol is capable of operating in three modes: as a primary server, as a secondary server, or as a client. The primary server is the one, which is synchronized directly to a reference clock. The secondary server has upstream and downstream servers or clients to synchronize. In order to maintain stability in large NTP subnets, secondary servers should be fully NTPv4-compliant. The client may synchronize upstream servers, but it does not provide synchronization between dependent clients.

Unfortunately, NTP server was not designed to run inside of a VM because it requires a high resolution system clock with response times to clock interrupts that are serviced with a high level of accuracy, and no known VM is capable of meeting these requirements [41]. Despite that NTP server faces problems in a VM, an NTP client can cope well with it. All the CSPs are aware of the time synchronization problem, and most of them propose the use of an NTP client. Amazon, for example, proposes the users on Linux instances to disable the hypervisor's wall-clock and then employing NTP.

W32TIME: The Windows version of NTP is called W32TIME. W32TIME uses the Simple Network Time Protocol [42], a simplified access strategy for servers and clients that do not require the degree of accuracy that NTP provides [43]. W32TIME is installed by default in all Windows based computers, and it uses the UTC. All Windows based CSP use W32TIME as their synchronization technique. Because UTC is independent of the time zone of a specific digital machine, time zone is stored in the machine's memory, and it is added to the system time just before it is displayed to the user.

The W32TIME service starts automatically when a computer joins a domain and at this time takes place the initial time synchronization. A special purpose service, called Net Logon, looks for an authenticated controller that can authenticate and synchronize time with the computer [39]. When it finds one, it starts a communication, exchanging Simple Network Time Protocol packets in order to calculate the time offset and the roundtrip delay between the two parts [43].

Every 45 min, the W32TIME connects to the authenticated controller in order to check for time drifts in the local clock. This will occur for three separate times, and in case there is no time drifting, or time drifting is humble, the next connection will occur at 8-hour intervals. In case of a failure, the whole process starts from the beginning. There are three configuration methods in W32TIME. The first one is the domain hierarchy based synchronization, the second is using a manually specified synchronization source, and the last one is using no synchronization at all. The domain hierarchy based synchronization is the default on every windows based machine.

The same techniques can be, also, adapted by the VM's guest OS for timekeeping. For instance, such an approach is used by the XEN hypervisor. More precisely, each guest OS runs its own synchronization protocol, in this case the NTP. That means that, the guest OS connects to its own network time server, and it is using its own flow on NTP timing packets [44]. This solution works well in terms of resource efficient use, but it is not the best one in terms of performance. This occurs because the additional latencies by the guest OS make the NTP pushed into instability [44]. The same method can be used when a VM migrates to another physical machine. Unfortunately, many and consecutively migration events cause time drifts of order of magnitude of seconds, and thus the method is considered to be ineffective.

6.2. Synchronization with a local time source

The guest OS, running inside a VM, can keep its wall-clock correct by synchronizing with the physical machine's local clock (Figure 2). All hypervisors offer a basic application for timekeeping, as a system call, but without ensuring time drifts. Usually, they monitor the

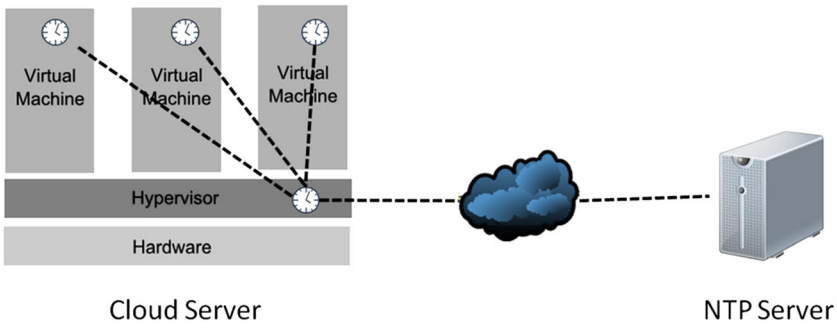

Figure 2. Synchronization with a local time source.

Security Comm. Networks 2016; **9**:571–582 © 2014 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

clock of each VM and compare it with the hosting machine's local clock. When a deviation is identified, the hypervisor either modifies the guest OS clock without any warning or just notifies the OS. The hypervisor can, also, take action when a VM was suspended, or a snapshot was restored. Next, we analyze in more details four hypervisors: Xen, VMware, Hyper-V, and Kernel-based virtual machine (KVM).

In a typical Xen VM, Dom0 can be the only VM that runs a full clock synchronization algorithm. In this scenario, timestamps from the NTP are communicated to DomU guests via the periodic adjustment of a boot time variable in the hypervisor [44]. Timestamping in DomU is achieved by a modified system clock call, which uses Xen Clocksource to extrapolate from the last time this variable was updated forward to the current time.

Virtual machine ware uses a patent-pending technique that allows the many timer devices in a VM to fall behind real time and catch up as needed [45]. According to VMware's information guide, it has been measured that these are sufficiently consistent with one another, so that software running in the VM is not disrupted by anomalous time readings.

Hyper-V offers also basic time synchronization services [46]. It achieves that by getting time readings from the underlying OS and sending them over to the guest OS. After receiving these time tokens, the guest OS delivers them to the Windows timekeeping infrastructure in the form of a Windows time provider. Also, these time tokens are adjusted for any time zone difference between the management operating system and the guest OS.

Finally, the KVM hypervisor provides a para-virtualized clock to its guest OSes [47]. Para-virtualized devices in general are drivers for virtual devices that increase the I/O performance of VMs [48]. Because guest OSes, which are using the time stamp counter (TSC) as a clock source, may suffer timing issues, KVM works around hosts that do not have a constant TSC by providing guests with a para-virtualized clock. Besides that, guest OSes are, also, able to use other ×86 clock sources for their timing synchronization.

7. TOWARD RELIABLE CLOUD TIMEKEEPING TECHNIQUES FOR CLOUD FORENSICS INVESTIGATION

7.1. The role of the cloud service provider

Practically, all the proposed solutions are strongly depending on the CSP. This is something that was expected given that all the virtual resources are totally under the CSP's control. In order to better evaluate the impact of this dependency, we can define two security models. In the first model, the CSP is honest, and in the second is malicious.

In the first attack model, the honest CSP is trying to offer the best service to its customers, using one or several reliable time sources and the best time possible synchronization protocol. In this scenario, the clients can depend on the CSP for accurate time provision. However, they have to trust CSP's capability to keep the physical machine's clock with the correct time. This is in line with all the risks that a user takes when he is using CC. CSP can be the single point of failure.

In the malicious scenario, there is no protection what so ever for the client. The CSP can manipulate the VM's internal clock, as easily it can control any of the virtual resources. Even when an external time server is used by the guest OS, the results are subject to the message delays that the hypervisor can artificially create.

7.2. The role of (trusted) time service providers

In most of the existing solutions, there is a strong dependency on the time providers. Either the CSP and/or the VM guest OS are using a network time protocol to synchronize with a network time server. Time correctness strongly depends on the reliability of this service. Obtaining the time from many servers and keeping the most common time value from them, is considered to be the best strategy. Time servers located far away must be avoided as possible, since the delays, like network flatten, are time varying and thus uncontrollable.

7.3. The role of the cloud computing client

Timekeeping is a very challenging task, and most of the protocols require complicated set up and management procedures. For instance, the NTP's accuracy is variable and strongly depends on the chosen parameters. If the clock needs to be accurate down to the millisecond instead of just to the second, the configuration can be very complicated. Also, the mobility actions like, VM's migration, suspension or storage, can lead to severe time drifts. It seems that leaving timekeeping totally at the CC client's hands premises technical skills on his behalf.

7.4. Time synchronization logging

The clock of an OS, no matter if we are dealing with the hypervisor or the guest OS, is one of the most valuable resources for a CF investigation, and it must be treated as so. Clock related log-records must be kept, by recording information like, when the clock was checked, who did the checking, which protocol was used, what was the result of the check. Any identified time drifts must be listed and any corrections must be reported.

7.5. Provable time correctness

Admissibility of evidence can be enforced with technical proofs. Nowadays, contemporary cryptography offers the tools for provable clock management. Any modification

of time, any application of time synchronization protocol can become non-repudiable using public key cryptographic techniques. However, these tools are not supported by the existing time synchronization techniques.

7.6. Evaluating time synchronization protocols

Network Time Protocol and W32TIME are not the only network-based time synchronization protocols. The well-known Precision Time Protocol (PTP) is another option for systems synchronization, while new CC oriented protocols have been also proposed, like the Robust Absolute and Difference Clock (RADclock). Next, we briefly present and evaluate all the options.

PTP: The PTP [49] is a protocol for clock synchronization throughout a computer network. It was originally defined in the IEEE 1588-2002 standard, and it was published in 2002. Its current version is v2.0, which was released in 2008. The PTP is mainly used to synchronize device clocks in special-purpose industrial automation and measurement networks [49]. These devices must be interconnected by switches in a LAN. PTP on-wire operates primarily in a broadcast mode. In this mode, a grandmaster provides synchronization to many clients on the same network. Except for broadcast mode, PTP can operate in master/slave mode, too. In this mode, the synchronization between the participants occurs with client/server message exchange. The accuracy expectation of PTP synchronized clocks is in the order of 100 ns determined primarily by the resolution and stability of a dedicated clock oscillator and counter [49].

RADclock: Recently, a new timekeeping architecture, especially designed for Xen environments, was introduced [44]. This is an architecture used exclusively for VM environments. It is built upon a feed-forward based synchronization algorithm, RADclock, and it is considered the successor of the NTP. So far, it is available for systems running FreeBSD and Linux. In this algorithm, timing packets are timestamped using raw packet timestamps. The clock error is estimated from these timestamps and the server timestamps. This is a feed-forward approach, because errors are corrected based on post-processing outputs, and these are not themselves fed back into the next round of inputs. That means that, the timestamps are independent from the clock state and therefore from potential time drifts. The basic idea is the use of one clock as a reference clock, that is, the system clock is considered as the reference clock, and all the VMs should use that.

Protocols comparison: Next, we evaluate the four protocols in terms of accuracy, supporting hardware, precision expectations, and application area. Our analysis appears in Table I. NTP, W32TIME and RADclock require no hardware provisions to capture timestamps, while PTP requires hardware provisions to capture timestamps. These hardware provisions are the major reason that makes PTP less popular than the others. PTP is used most often to synchronize device clocks in special-purpose industrial automation and measurement networks. NTP is used most often to synchronize system clocks in general-purpose workstations and servers.

Regarding the supported accuracy, NTP has no constant accuracy. Its accuracy depends totally on the underlying hardware. On the other side, PTP's accuracy is of the magnitude of 100 ns, W32TIME's accuracy is in milliseconds, and RADclock's accuracy is below 1 μs. Also, NTP and W32TIME's timescale is in seconds, PTP's timescale is in nanoseconds, and RADclock timescale is milliseconds.

In terms of usage, the NTP, W32TIME, and RADclock protocols mainly appear in ordinary computer systems in order to maintain their time up to date. From the other side, PTP is used in devices which operate in industrial automation because of its accuracy. Finally, regarding the offered security, almost all of them offer message integrity protection and authentication.

7.7. Discussion

Digital forensic processes need to be developed or reviewed in order to meet the needs of CC investigations. However, the deployment of instruments/models to enable timestamp synchronization has to respect the principle that the collected evidence should be an authentic representation of the events investigated. That means that, timestamp synchronization should be conducted in a way to avoid any intentional or not spoliation or alteration of evidence or simply been regarded as such by the court as in such a case it would lead to inadmissibility of the evidence. The three security requirements that have been identified for the log-files must be satisfied for the timestamping procedure as well.

Toward the design of a timekeeping system that reliably a CF investigator can use, further research is needed. More precisely, as a direct outcome of our analysis in the previous sections, we have compiled the following list of recommendations and guidelines.

- New time synchronization protocols especially for CC must be designed.
- Clients must regain control over their computations. New cryptographic tools are needed for secure computation outsourcing, including time counting.
- The client must trust the CSP for timekeeping when he does not possess the necessary technical skills that accurate time synchronization requires.
- All operations on a clock must be logged. Logging must guarantee non-repudiation of the log-records.
- All operations on the clock must be secure, that is, the entities that participate in the time synchronization protocol must be authenticated, and the exchanged messages must be protected in terms of integrity.
- It is most efficient for a VM to use the clock of the physical machine than to synchronize with a remote server.
- A policy must describe in details all the operations that the CSP and the guest VM must perform in terms of clock synchronization.

Table I. Comparison of the existing techniques.

	NTP	PTP	RADclock	W32TIME
Supporting hardware	No	Yes	No	No
Accuracy	Not stable	100 ns	$<1\mu s$	ms
Security	Symmetric and public key	Symmetric	Not available	Symmetric
	Integrity authentication	Integrity authentication	Not available	Integrity authentication
Application Area	System clocks	Industrial automation	System clocks	System clocks

NTP, Network Time Protocol; PTP, Precision Time Protocol; RADclock, Robust Absolute and Difference Clock.

- Policymakers must take into consideration that log-file's will be used in a CF investigation and that they must be admissible in a court as evidence. Because, there is always a trade-off between satisfying this requirement and maximizing the system's performance, an equilibrium point must be found.
- Several external time servers must be used for accurate time synchronization. Server's distance and quality of service must be taken into account.

8. CONCLUSION

In this paper, we have investigated the problem of timekeeping in CC, and its impact in CF investigations. Our analysis is focused on the accuracy that log-records timestamping using contemporary time synchronization techniques. It has been identified that further research is required, as the nature of CC does not fully comply with existing time synchronization, jeopardizing the admissibility in a court of law of crucial evidence like the log-file's contents.

New timekeeping techniques that take into consideration the special features of CC must be designed. In this line, we have compiled a list of guidelines toward the design of CF aware timekeeping techniques for CC. We expect that the next generation of CC platforms will support CF by design.

REFERENCES

1. 2012 Norton CyberCrime Report. *Symantec Corp.*, 2012.
2. Strategy sets out common EU approach to securing digital networks, preventing online crime and protecting consumers. *Cyber-Crime Blueprint, European Commission*, 2013.
3. Zhou W, Sherr M, Marczak WR *et al*. Towards a data centric view of cloud security. *2nd International Workshop on Cloud Data Management*, 2010; 25–32.
4. Carrier B. Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of Digital Evidence* 2003; 1–12.
5. Biggs S, Vidalis S. Cloud computing storms. *International Journal of Intelligent Computing Research IJIC Review* 2010; **1**(1): 63–72.
6. Biggs S, Vidalis S. Cloud computing: the impact on digital forensic investigations. *Paper presented at the 4th International Conference for Internet Technology and Secured Transactions*, London, United Kingdom, 2009.
7. Zimmerman S, Glavach D. Cyber forensics in the cloud. *Ianewsletter* 2011; **14**(01): 4–7.
8. Ruan K, Carthy J, Kechadi T, Crosbie M. Cloud forensics: an overview. *Advances in Digital Forensics VII*, 2011.
9. Kenneally E. Digital logs – proof matters. *Digital Investigation* 2004; **1**(2): 94–101.
10. Mell P, Grance T. The NIST definition of cloud computing. *National Institute of Standards and Technology*, October 7, 2009.
11. Arasteh AR, Debbabi M, Sakha A, Saleh M. Analyzing multiple logs for forensic evidence. *Digital Investigation*, **4**; 2007: 82–91.
12. Saleh M, Arasteh AR, Sakha A, Debbabi M. Forensic analysis of logs: modeling and verification. *Knowledge-Based Systems* 2007; **20**(7): 671–682.
13. Takahashi D, Xiao Y. Complexity analysis of retrieving knowledge from auditing log files for computer and network forensics and accountability. *Proc. of IEEE ICC '08*, 2008; 1474–1478.
14. Forte DV. The "art" of log correlation. *Proceedings of the Information Security South Africa (ISSA),* Enabling Tomorrow Conference 2004.
15. Marty R. Cloud application logging for forensics. *Symposium on Applied Computing ACM '11*, TaiChung, Taiwan, 2011.
16. Log analyzer for Amazon CloudFront. *Amazon Web Services*, 2009.
17. Ko RKL, Jagadpramana P, Lee BS. Flogger: a file-centric logger for monitoring file access and transfers within cloud computing environments. *HP Labs Singapore*, Tech. Rep. 2011; HPL-2011-119.
18. Zawoad S, Dutta AK, Hasan R. SecLaaS: secure logging-as-a-service for cloud forensics. *Proceedings of the 8th ACM SIGSAC symposium on Information*, computer and communications security, Asia CCS '13, 2013; 219–230.
19. Wongthai W, Rocha FL, Moorsel AV. A generic logging support architecture for infrastructure as a service (IaaS) cloud. *School of Computing Science*, University of Newcastle upon Tyne, 2012.

20. SANS consensus project information system audit logging requirements. Information Systems Audit Logging, SANS Institute, 2007.
21. Taylor M, Haggerty J, Gresty D, Hegarty R. Digital evidence in cloud computing systems. *Computer Law & Security Review* 2010; **26**(3): 304–308. doi:10.1016/j.clsr.2010.03.002.
22. Peisert SP. A model of forensic analysis using goal-oriented logging. *PhD thesis,* Department of Computer Science and Engineering, University of California, San Diego, 2007.
23. Balboni P, Pelino E. Law enforcement agencies' activities in the cloud environment: a European legal perspective. *Information & Communications Technology* 2013; **22**(2): 165–190.
24. Kelley D. How data-centric protection increases security in cloud computing and virtualization. *Security Curve,* 2011.
25. Ko RKL, Jagadpramana P, Mowbray M, *et al*. TrustCloud - a framework for accountability and trust in cloud computing. *Proc. IEEE 2nd Cloud Forum for Practitioners, (IEEE ICFP '11)*, IEEE Computer Society, 2011; 1–5.
26. Ko RKL, Lee BS, Pearson S. Towards achieving accountability, auditability and trust in cloud computing. *Proc. International workshop on Cloud Computing: Architecture, Algorithms and Applications (CloudComp2011)* Athens, Greece, Springer, 2011; pp. 5.
27. Accorci R. Safekeeping digital evidence with secure logging protocols state of the art and challenges. *Proceedings IMF '09,* 2009; 94–110.
28. Lonvick C. The BSD syslog protocol. RFC 3164, IETF, 2001.
29. Kent K, Souppaya M. Guide to computer security log management. *Recommendations of the National Institute of Standards and Technology,* 2006.
30. Schneier B, Kelsey J. Secure audit logs to support computer forensics. *ACM Transactions on Information and System Security (TISSEC)* 1999; **2**(2): 159–176.
31. Karyda M, Mitrou L. Internet forensics: legal and technical issues. *Digital Forensics and Incident Analysis,* WDFIA 2007. Second International Workshop on. IEEE, 2007; 3–12.
32. Taylor M, Haggerty J, Gresty D, Lamb D. Forensic investigation of cloud computing systems. *Network Security* 2011; **2011**: 4–10.
33. Reilly D, Wren C, Berry T. Cloud computing: forensic challenges for law enforcement. *Internet Technology and Secured Transactions (ICITST)*, 2010 International Conference for IEEE, 2010; 1–7.
34. Grispos G, Storer T, Glisson WB. Calm before the storm: the challenges of cloud. *Emerging Digital Forensics Applications for Crime Detection, Prevention, and Security* 2013; **4**: 28–48.
35. Ruan K, Baggili I, Carthy J, Kechadi T. Survey on cloud forensics and critical criteria for cloud forensic capability: a preliminary analysis. *Proceedings of the 2011 ADFSL Conference on Digital Forensics*, Security and Law, 2011.
36. Adams CW. Legal issues pertaining to the development of digital forensic tools. *Systematic Approaches to Digital Forensic Engineering,* SADFE'08. *Third International Workshop on. IEEE*, 2008; 123–132.
37. Porcedda MG. Law enforcement in the clouds: is the EU data protection legal framework up to the task? *European Data Protection: In Good Health?* Gutwirth S, Leenes R, De Hert P, Poullet Y (eds). Springer, 2012; 203–232.
38. Hooper C, Martini B, Raymond Choo KK. Cloud computing and its implications for cybercrime investigations. *Australia Computer Law & Security Review* 2013; **29**(2): 152–163.
39. Boyd C, Forster P. Time and date issues in forensic computing – a case study. *Digital Investigation* 2004; **1**: 18–23.
40. Mills D. Internet time synchronization: the Network Time Protocol. *IEEE Transactions on Communication* 1991; **39**: 1482–1493.
41. How do I establish clock synchronization in the cloud (AWS, heroku, etc) across many nodes? StackOverFlow. Available at http://stackoverflow.com/questions/8743002/how-do-i-establish-clock-synchronization-in-the-cloud-aws-heroku-etc-across, [Accessed on August 5, 2013]
42. Brandolini S, Green D. *The Windows Time Service*. Microsoft Corp: Redmont, USA, 2001.
43. Mills D. Simple Network Time Protocol (SNMP) Version 4 for IPv4, IPv6 and OSI. *Internet Engineering Task Force (IETF),* RFC2030, RFC1769, 1996.
44. Broomhead T, Cremean L, Ridoux J, Veitch D. Virtualize everything but time. *Proc. OSDI '10*, Vancouver, Canada, 2010.
45. Timekeeping in VMware virtual machines. *Information Guide*, VMWARE, 2010.
46. Armstrong B. Time synchronization in hyper-V. *Ben Armstrong Virtualization Blog,* 2010.
47. KVM guest timing management. *Fedora Documentation*, Chapter 17.
48. Para-virtualized devices. *Virtualization Getting Started Guide*, RedHat Product Documentation.
49. Rodrigues S. IEEE 1588 and synchronous Ethernet in telecom. In *Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement*, Control and Communication ISPCS 2007, Gaderer G, Lee K (eds). IEEE: Vienna, Austria, 2007; 138–142.