# Secure Evidence Collection and Storage for Cloud Accountability Audits

Thomas Ruebsamen[1], Tobias Pulls[2] and Christoph Reich[1]
[1]*Cloud Research Lab, Furtwangen University, Furtwangen, Germany*
[2]*Department of Mathematics and Computer Science, Karlstad University, Karlstad, Sweden*
{*thomas.ruebsamen, christoph.reich*}*@hs-furtwangen.de, tobias.pulls@kau.se*

Keywords: Cloud Computing, Security, Accountability, Digital Evidence.

Abstract: Cloud accountability audits can be used to strengthen trust of cloud service customers in cloud computing by providing reassurance regarding the correct processing of personal or confidential data in the cloud. However, such audits require various information to be collected. The types of information range from authentication and data access logging to location information, information on security controls and incident detection. Correct data processing has to be proven, which immediately shows the need for secure evidence record storage that also scales with the huge number of data sources as well as cloud customers. In this paper, we introduce Insynd as a suitable cryptographic mechanism for storing evidence for accountability audits in our previously proposed cloud accountability audits architecture. We present our reasoning for choosing Insynd by showing a comparison of Insynd properties with requirements imposed by accountability evidence collection as well as an analysis how security threats are being mitigated by Insynd. Additionally, we describe an agent-based evidence collection process with a special focus on security and privacy protection.

## 1 INTRODUCTION

Cloud Computing is known for its on demand computing resource provisioning and has now become mainstream. Many businesses as well as private individuals are using cloud services on a daily basis. The nature of these services varies heavily in terms of what kind of information is being out-sourced to the cloud provider. More often than not that data is sensitive, for instance when Personal Identifiable Information (PII) is being shared by an individual. Also, businesses that move (parts of) their processes to the cloud, for instance by using a Customer Relationship Management Software as a Service provider, are actively participating in a major paradigm shift from having all data on-premise to moving data to the cloud.

New challenges come along with this trend. Two of the most important issues are customer trust and compliance (Jansen and Grance, 2011; Pearson, 2011). These issues are closely tied to the loss of control over data. When moving to the cloud, direct control over i) where data is stored, ii) who has access to it and iii) how it is shared and processed is given up.

Because of this loss of control, cloud customers have to trust cloud providers that they treat their data in an appropriate and responsible way. This includes providing information about data locality, isolation, privacy controls and data processing in general. One way to enable that trust is by strengthening transparency and accountability (Haeberlen, 2009; Weitzner et al., 2008) of the cloud provider and services.

To regain information on the kind of data processing, cloud audits can be used to check how it has been done. An important part of cloud audits is evidence collection. Depending on the data processing policies in place, various sources of evidence need to be considered. Logs are a very important source of evidence, when it comes to auditing the cloud operation (e.g., access logs and error logs). However, other sources of information are also important, such as files or events registered in the cloud management system. To capture evidence from this variety of sources, centralized logging mechanisms are not enough. We therefore propose a system for accountability evidence collection and audit. With this system, cloud providers are enabled to demonstrate their compliance with data handling policies to their customer's and third-party auditors in an automated way.

In our previous work, we introduced a system (Ruebsamen and Reich, 2013) for cloud accountability audits, that enables automated collection of evidential data in the cloud ecosystem with the goal of

performing accountability audits. A key mechanism of this system is the secure and privacy-friendly collection and storage of evidence. In our previous work we also explored the use of a somewhat homomorphic encryption scheme to secure evidence collected in the evidence store (Lopez et al., 2014). In this paper, we present a more practical alternative that imposes less restrictions on evidence collection. The contributions of this paper are:

- An architecture for automated evidence collection for the purpose of cloud accountability audits
- A process for secure and privacy-protecting evidence collection and storage

The remainder of this follow-up paper is structured as follows: in Section 2 we present related work in the area of secure evidence collection and cloud auditing. The core principles of Insynd are introduced in Section 3. Following that, we present in Section 4 a mapping of typical characteristics of digital evidence and secure evidence collection in the cloud to how these are addressed by integrating Insynd in our audit agent system. In Section 5 we describe the architectural details of the Insynd integration. We present a scenario-based informal evaluation of our system in Section 6 and conclude this paper in Section 7.

## 2 RELATED WORK

Redfield and Date propose a system called Gringotts (Redfield and Date, 2014) that enables secure evidence collection, where evidence data is signed at the system that produces it, before it is sent to a central server for archival using the Evidence Record Syntax. It is similar to our system with respect to the automatic collection of evidential data from multiple sources. However, their focus is on the archival of evidence, whereas we propose a system that also enables automated evidence processing for audits. Additionally, our system also addresses privacy concerns of evidence collection in a multi-tenant environment such as the cloud by introducing evidence encryption, whereas Redfield and Date focus on archival and preservation of evidence integrity.

Zhang et al. (Zhang et al., 2013) identify potential problems when storing massive amounts of evidential data. They specifically address possible information leaks. To solve these issues, they propose an efficient encrypted database model that is supposed to minimize potential data leaks as well as data redundancy. However, they focus solely on the storage backend and do not provide a workflow that addresses secure evidence collection as a whole.

Gupta (Gupta, 2013) identifies privacy issues in the digital forensics process, when it comes to data storage devices that typically do not only contain investigation related data, but may also hold sensitive information that may breach privacy. He also identifies a lack of automation in the digital investigation process. To address these issues, Gupta proposes the Privacy Preserving Efficient Digital Forensic Investigation (PPEDFI) framework. PPEDFI automates the investigation process by including knowledge about previous investigation cases, and which kinds of files were relevant then. With that additional information, evidence search on data storage devices is faster. However, while Gupta acknowledges privacy issues, the PPEDFI framework is focused on classic digital forensics and may not be applicable to a cloud ecosystem, where there is typically no way of mapping specific data objects to storage devices, in full.

The Security Audit as a Service (SAaaS) system proposed by Doelitzscher et al. (Doelitzscher et al., 2012; Doelitzscher et al., 2013) is used to monitor cloud environments and to detect security incidents. SAaaS is specifically designed to detect incidents in the cloud and thereby consider the dynamic nature of such ecosystems, where resources are rapidly provisioned and removed. However, the main focus of SAaaS is not to provide auditors with a comprehensive way of auditing the cloud provider's compliance with accountability policies, which requires additional security and privacy measures to be considered in the data collection process.

## 3 INSYND

Insynd is a cryptographic scheme where a forward-secure *author* sends messages intended for *clients* through an untrusted *server* (Pulls and Peeters, 2015b; Pulls and Peeters, 2015a; Pulls et al., 2013). The author is forward-secure in the sense that the author is initially trusted but assumed to turn into an active adversary at some point in time (Bellare and Yee, 2003). Insynd protects messages sent prior to author compromise. The server is completely untrusted, which is possible thanks to the use of Balloon, a forward-secure append-only persistent authenticated data structure (Pulls and Peeters, 2015a). This means that the server storing all messages can safely be outsourced, e.g., to traditional cloud services. Clients are assumed trusted to read messages sent to them by authors. Insynd contains support for clients to also be in the forward-security model, by discarding key-

material as messages are read.

Insynd provides the following properties:

**Forward Integrity and Deletion Detection.** Nobody can modify or delete messages sent prior to author compromise, as defined by Pulls et al. (Pulls et al., 2013). This property holds independently for Balloon (the data structure) and the Insynd scheme. For Balloon, anyone can verify the consistency of the data structure, i.e., it is publicly verifiable (Pulls and Peeters, 2015a).

**Secrecy.** Insynd provides public-key authenticated encryption (An, 2001) thanks to the use of NaCl (Bernstein et al., 2012).

**Forward Unlinkability of Events.** For each run by the author of the protocol to send new messages, all the events sent in that run are unlinkable. This implies that, e.g., an attacker (or the server) cannot tell which events belong to which client (Pulls and Peeters, 2015b). When clients receive their events by querying the server, if they take appropriate actions including but not limited to accessing the server over an anonymity network like Tor (Dingledine et al., 2004), their events remain unlinkable.

**Publicly Verifiable Proofs.** Both the author and client receiving a message can create publicly verifiable proofs of the message sender (the author), the receiving client (by registered identity), and the time the message was sent relative to e.g. a time-stamping authority (Pulls and Peeters, 2015b). The proof-of-concept implementation of Insynd uses Bitcoin transactions (Nakamoto, 2008) as a distributed time-stamping server.

**Distributed Settings.** Insynd supports distributed authors, where one author can enable other authors to send messages to clients it knows of without requiring any interaction with clients. Client identifiers (public keys) are blinded in the protocol, ensuring forward-unlinkable client identifiers between different authors (Pulls and Peeters, 2015b).

Pulls and Peters show that Insynd provides forward integrity and deletion detection, secrecy, publicly verifiable proofs, and forward-unlinkability of client identifiers in the standard model under the assumptions of the decisional Diffie-Hellman (DDH) assumption on Curve25519, an unforgeable signature algorithm, an unforgeable MAC, a collision and pre-image resistant hash function, and the security of the time-stamping mechanism (in our case, the Bitcoin block-chain) (Pulls and Peeters, 2015b). Forward unlinkability of events is provided in the random oracle model under the DDH assumption on Curve25519 (Pulls and Peeters, 2015b). The prototype implementation of Insynd shows performance comparable to state-of-the-art secure logging schemes, like PillarBox (Bowers et al., 2014), securing syslog-sized messages (max 1KiB) in the order of hundreds of microseconds on average on a commodity laptop. We stress that Insynd is subject to its own review and evaluation; in this paper, we use Insynd as a building block to facilitate secure evidence collection and storage for cloud accountability audits.

# 4 AUDIT EVIDENCE STORAGE REQUIREMENTS

In this Section, we present a comparison of general evidence attributes, how they apply in the context of evidence collection for cloud accountability audits and how the integration of Insynd solves key issues in evidence storage.

## 4.1 Requirements of Digital Evidence

In (Mohay et al., 2003) the core principles of any evidence are described as:

**Admissibility.** Evidence must conform to certain legal rules, before it can be put before a jury.

**Authenticity.** Evidence must be tieable to the incident and may not be manipulated.

**Completeness.** Evidence must be viewpoint agnostic and tell the whole story.

**Reliability.** There cannot be any doubts about the evidence collection process and its correctness.

**Believability.** Evidence must be understandable by a jury.

These principles apply to common evidence as well as digital evidence. Therefore, the evidence collection process for audits has to consider special requirements, which help in addressing these attributes and ensure best possible validity in audits and applicability in court.

In Table 1 we present a mapping of the previously described evidence attributes and how they are supported by the integration of Insynd as a means of storing evidence records. We thereby focus on the key properties of Insynd as described in Section 3.

*Admissibility* of digital evidence is influenced by the transparency of the collection process and data protection regulation. Digital evidence can be any kind of data (e.g., e-mail messages, social network messages, files, logs etc.). Insynd does not have any

direct influence on the admissibility of the evidence stored in it.

*Authenticity* of digital evidence before court is closely related to the integrity requirement put on evidence records. Evidence may not be manipulated in any way and must be protected against any kind of tampering (willingly and accidentally). Insynd ensures that data cannot be tampered with once it is stored.

*Completeness* is not directly ensured by Insynd, but rather needs to be ensured by the evidence collection process as a whole. Especially important are the definition of which evidence sources provide relevant evidence that need to be considered during the collection phase. Insynd can complement the evidence collection process by providing assurance of that all data stored in the evidence store are made available as evidence, and not cherry-picked.

*Reliability* is indirectly supported by integrating necessary mechanisms into the evidence collection process, such as Insynd.

*Believability* of the collected evidence is not influenced by implemented mechanisms, but rather by the interpretation and presentation by an expert in court. This is due to judges and juries usually being non-technical, which requires an abstracted presentation of evidence. Insynd does not influence the believability in that sense.

Table 1: Mapping the Impact of Insynd Properties to Evidence Attributes.

| | | Insynd | |
|---|---|---|---|
| | | Forward Integrity and Deletion Detection | Publicly Verifiable Proofs |
| ES | Admissibility | | |
| | Authenticity | ✓ | ✓ |
| | Completeness | ✓ | ✓ |
| | Reliability | ✓ | ✓ |
| | Believability | | |

## 4.2 Privacy Requirements

Not all requirements that a secure evidence storage has to fulfill can be captured by analyzing the attributes of digital evidence. Other aspects have to be taken into account to address privacy concerns. Protecting privacy in the process of evidence collection is utmost importance, since the collected data is likely to contain personal data. For cloud computing, one limiting factor may be whether or not the cloud provider is willing to provide deep insight into its infrastructure. Table 2 presents a mapping of privacy principles and properties of our evidence process.

Below we summarise some key privacy principles:

**Confidentiality.** of data evolves around mechanisms for the protection from unwanted and unauthorized access. Typically, cryptographic concepts, such as encryption, are use to ensure confidentiality of data.

**Data Minimization.** states that the collection of personal data should be minimized and limited to only what is strictly necessary.

**Purpose Binding.** of personal data entails that personal data should only be used for the purposes it was collected for.

**Retention Time.** is concerned with how long personal data may be stored and used, before it needs to be deleted. These periods are usually defined by legal and business requirements.

Insynd and our evidence process provides various mechanisms that support these privacy principles.

*Confidentiality* A central property of Insynd is that it is always encrypting data using public-key cryptography. By encrypting the evidence store, compromising the privacy of cloud customer data that has been collected in the evidence collection processes becomes almost impossible by attacking the evidence store directly. This goes as far as being able to safely outsource the evidence store to an untrusted third-party, a key property of Insynd (Pulls and Peeters, 2015b).

*Data Minimisation* Furthermore, Insynd provides forward unlinkability of events and client identifiers, as described in Section 3, which helps prevent several types of information leaks related to storing and accessing data. Collection agents are always configured for a specific audit task, which is very limited in scope of what needs to be collected. Agents are never configured to arbitrarily collect data, but are alway limited to a specific source (e.g., a server log) and data objects (e.g., a type of log events).

*Purpose Binding* Neither Insynd nor our evidence process can directly influence the purpose for which collected data is used. Indirectly, the use of an evidence process like ours, incorporating secure evidence collection and storage, may serve to differentiate data collected for auditing purposes with other data collected e.g., for marketing purposes.

*Retention time* poses a real challenge. In cloud computing, the precise location of a data object is usually not directly available, i.e., the actual storage medium used to store a particular block is unknown, making data deletion hard. However, if data has been

encrypted before storage, a reasonably safe way to ensure "deletion" is to discarding the key material required for decryption. Insynd supports forward-secure clients, where key material to decrypt messages are discarded as messages are read.

Table 2: Mapping of Insynd properties to Evidence Collection Requirements.

| | | Insynd | | |
|---|---|---|---|---|
| | | Secrecy | Forward Unlinkability of Events | Forward Unlinkability of Recipients |
| ES | Confidentiality | ✓ | ✓ | ✓ |
| | Data Minimisation | | ✓ | ✓ |
| | Purpose Binding | | | |
| | Data Retention | ✓ | | |

In Section 6, we also describe the threat model for the system described in this paper and present an evaluation of how Insynd is used to mitigate these threats.

# 5 SECURE EVIDENCE STORAGE ARCHITECTURE

In this Section, we provide an architectural overview of the integration of Insynd into a secure evidence collection and storage process. We describe the overall architecture and its components, how the components of Insynd are mapped into the audit agent system and which setup process is required to use Insynd for securing evidence collection and storage.

## 5.1 Architecture

In this Section we discuss the architectural integration of Insynd as an evidence store in our audit system. There are basically three different components required to perform secure evidence collection. Figure 1 shows an overview of these components - *Evidence Source*, *Evidence Store* and *Evidence Processing* - as well as the flow of data between them. From the various sources of evidence in the cloud, evidence records are collected that will be stored in the evidence store on a per-tenant basis. The evidence store is thereby located on a separate server. As previously mentioned, the server may be an untrusted third-party cloud storage provider. This is important to ensure so that this approach scales well with a growing number of tenants, evidence sources and evidence records.

Our architecture is built around using software agents for evidence collection, evidence evaluation and controlling the overall system. Agent technology helps with extensibility by allowing us to easily introduce new evidence sources and processors by building new agents. On top of that, it allows the audit system to address rapid infrastructure changes, which are very common in cloud infrastructures by easily deploying and destroying agents when needed. We base our system on the Java Agent DEvelopment Framework (JADE, 2015). This effectively means that anywhere, where a Java runtime environment is available, a collection agent can be deployed.

### 5.1.1 Evidence Collection

There are various evidence sources to be considered, such as logs, cryptographical proofs, documentation and many more. For each, there needs to be a suitable collection mechanism. For instance, a log parser for logs, a tool for cryptographical proofs or a file retriever for documentation. This is done by a software agent called *Evidence Collection Agent* that is specifically developed for the data collection from the corresponding evidence source. The collection agent acts as an *Insynd Author* meaning it uses the *Sender API* to store evidence into the Evidence Store. The encryption happens in the Sender API. Typically, this agent incorporates or interfaces with a tool to collect evidential data, for instance forensic tools, such as file carvers, log parsers or simple search tools. Another type of collection agent have client APIs implemented to interface with more complex tools, such as Cloud Management Systems (CMS). Generally, these agents receive or collect information as input and translate that information into an evidence record, before storing it in the Evidence Store.

### 5.1.2 Evidence Storage

From the Evidence Collection Agent, evidence records are sent to the Evidence Store. The Evidence Store is implemented by the *Insynd Server*. Since Insynd functions as a key-value store for storing evidence records (encrypted messages identified by a key) NoSQL or RDBMS-based backend for persisting evidence records can be used. All data contained in the Evidence Store is encrypted. Each record is addressed to a specific receiver (e.g., an Evidence Processing Agent). The receiver's public key is used in

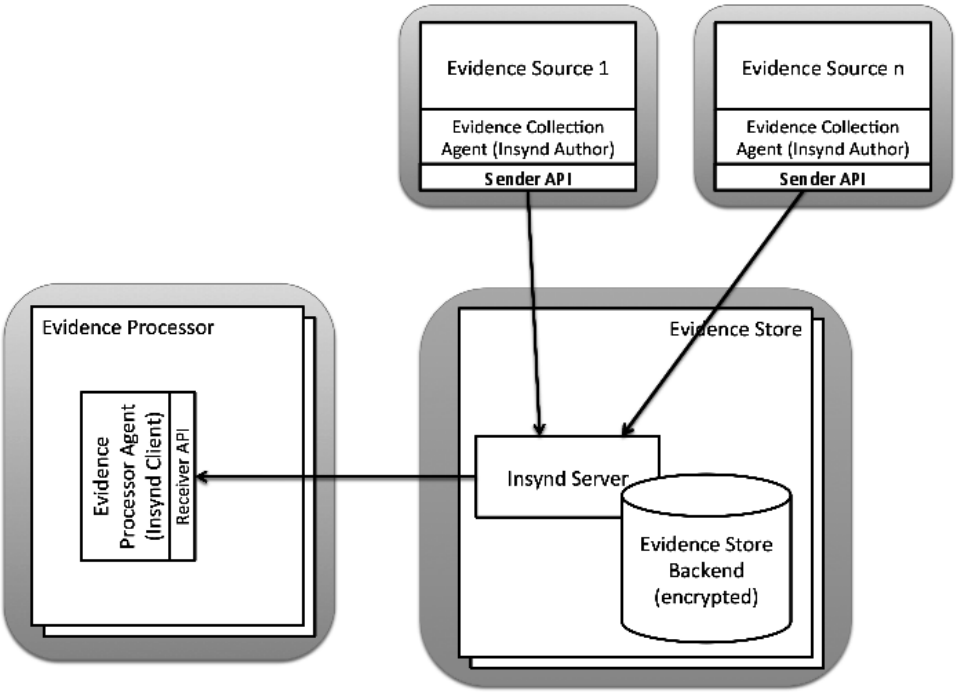


Figure 1: Evidence Collection, Storage and Processing Workflow.

the Sender API to encrypt the record on the Evidence Store. This means that only the receiver is able to access the evidence data from the Evidence Store. Isolation between tenants in a single Evidence Store is achieved by providing one container for each tenant where his evidence records are stored. However, even stronger isolation is also possible by providing a separate Evidence Store hosted on a separate VM. Additionally, Evidence records require a unique identifier in the Evidence Store to enable selective retrieval of records. In our implementation, we use a combination of a policy identifier and a rule identifier (where a rule is part of a policy) to enable the receiver to reduce the amount of records to receive to a manageable size.

#### 5.1.3 Evidence Processing

Evidence Processing components are located at the receiving end of this workflow. The Receiver API is used by the processing agent (Insynd Client) to retrieve evidence records from the Evidence Store. The receiver can request multiple records from a period of time at once. The Client is also in possession of the corresponding private key to decrypt evidence records, which means records can only be decrypted at the Client.

### 5.2 Identity Management and Key Distribution

Since asymmetric encryption is such an important part of our system, we describe the encryption key distribution sequence next. In this software agent-based system, the automated setup of key material and registration with Insynd is particularly important. Figure 2 depicts the initialization sequence of collection and processing agents with a focus on key distribution.

In Figure 2 we introduce an additional component beyond those already described in the general architecture: the *Controller*. The Controller serves as an entry point that controls the agent setup and distribution process in the audit system. It is an important part of the lifecycle management of the system's agents (e.g., creating and destroying of agents or migration between platforms).

In Figure 2 we describe the initialization sequence for a simple scenario, where a particular tenant wishes to audit compliance with a policy and one rule included in that policy in particular. The following steps have to be performed to setup the evidence collection and storage process for that particular rule:

1. In the first step, a Processing Agent is created and configured according to the input policy and rule respectively for the tenant.

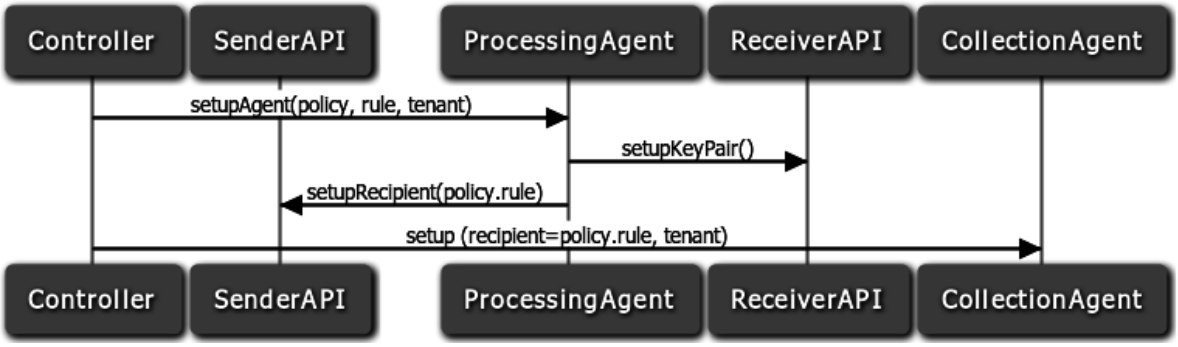


Figure 2: Evidence Collection Setup Sequence.

2. During the setup phase, the Processing Agent sets up a keypair at the Receiver API. The Receiver API is a RESTful service that holds private key material and is therefore located at the same servers hosting the Processing Agents (i.e., a trusted environment).
3. After the key material has been generated, the Processing Agent registers itself as a recipient at the Sender API. For this, it uses a unique identifier generated from the policy ID and the rule ID (i.e., *policyID.ruleID*).
4. In the last step, the Controller sets up the required Collection Agents and connects them with the corresponding Processing Agents by using the unique recipient identifier.

Now, it is possible for the Collection Agents to send evidence records to their corresponding Processing Agents. The messages will be encrypted at the Sender API service before storage, using the provided recipient's public key. The Processing Agent then pulls the evidence records from the Evidence Store using the Receiver API the records are decrypted using the receiver's private key.

# 6 EVALUATION

In this Section we present an informal security evaluation of the system we have implemented for secure evidence collection. We describe the evidence collection work flow using a fictitious scenario. By applying the evidence collection and storage process to the setting described in this scenario, we demonstrate how the requirements stated in Section 4 are addressed. Additionally, we provide a model that states threats and adversaries to the process as well as the mitigation functions introduced by Insynd.

In this scenario, the CCOMP company is a customer of the Infrastructure as a Service provider CloudIA. In particular, we analyze the security properties of the evidence collection process by looking at the data at rest as well as the data in transit protection at any time during the flow from the evidence source to its processor. We thereby assume that CloudIA is using OpenStack (OpenStack, 2015) as a its Cloud Management System (CMS), since this a widely popular open source CMS, which we use for developing our audit agent system. However, any other CMS could be used as well as long as it provides the needed monitoring interfaces.

## 6.1 Scenario

CloudIA is specialized in providing its customers with virtualized resources in the form of virtual machines, networks and storage. CCOMP has outsourced most of its IT services to CloudIA. Among them is a service that processes data of CCOMP's customers. For that data, CCOMP has to guarantee data retention. CCOMP has identified snapshots to be one major problem with respect to the data retention policy, since the virtual machine's storage is duplicated in the process. This means for CCOMP that in order to be compliant with the data retention policy, a snapshot of that virtual machine may have a maximum lifetime of one day, which limits its usefulness to e.g., backing up before patching. Now, we assume a trustworthy but sloppy administrator at CCOMP who creates a snapshot before patching software on the virtual machine, but then omits deleting the snapshot after he is done. However, an automated daily audit of its cloud resources was put in place by CCOMP to detect such compliance violations.

## 6.2 Implementation

The collection agent required for the above scenario communicates with our OpenStack CMS to gather evidence of the CMS behavior regarding virtual ma-

chine snapshots. The processing agent contains the logic for detecting snapshot violations (i.e., base virtual machine and a maximum age of the snapshot derived from the retention policy). The collection agent is deployed at the CMS controller node and has access to OpenStack's RESTful API. The processing agent is located on the same trusted host as the controller agent (see Figure 1 for reference). The evidence store is located on a separate, untrusted virtual machine. Now, the following steps are performed:

1. The collection agent opens a connection to the OpenStack RESTful API on the same host and requests a history of snapshot events for CCOMP's virtual machine. Despite there being no communication over the network, HTTPS is used to secure the communication between the collection agent and the CMS. Since the policy only requires information about snapshots to be collected, the CMS agent limits evidence record generation to exactly that information, nothing more.
2. The collection agent sets up the receiver of the evidence according to the process depicted in Figure 2 and sends the collected records to the evidence store (Insynd). The communication channel is encrypted using HTTPS and the payload (evidence records) is encrypted with the receiving agent's public key.
3. The processing agent pulls records from the evidence store in regular intervals (e.g., every 24 hours), analyses them and triggers a notification of a detected violation. The communication between the processing agent and the evidence store is secured using HTTPS.
4. In the last step, evidence records are deleted because their retention limit has been reached. This is done by discarding the keys required for decryption.

## 6.3 Threat Model

To demonstrate which security threats exist for the evidence collection process and Insynd is used to mitigate them, we describe the threat model for this system categorized according to the STRIDE(Microsoft Developer Network, 2015) threat categorization:

- **S**poofing Identity
- **T**ampering with Data
- **R**epudiation
- **I**nformation disclosure
- **D**enial of Service
- **E**levation of Privilege

We have identified the following major threats to the evidence collection and storage process:

- *Unauthorized access to evidence (S,I)*: the protection of evidence from being accessed by unauthorized persons. Possible adversaries are a malicious third-party evidence storage provider (cloud service provider), another tenant (isolation failure) or an external attacker. Using Insynd for evidence collection and storage addresses this threat since recipients of messages are authenticated using appropriate mechanisms such as user credentials for API authentication and public keys for encryption.
- *Data leakage (S,I)*: the protection from unintentional data leakage. This could be caused by misconfiguration (e.g., unencrypted evidence being publicly available). Using Insynd for evidence collection and storage addresses this threat by encrypting data by default.
- *Eavesdropping, (T,I)*: the protection of evidence during the collection phase, especially in transit. Possibly adversaries are another tenant (isolation failure) or external attackers in case evidence is transported to an external storage provider or auditor. Using Insynd for evidence collection and storage addresses this threat by using transport layer as well as message encryption.
- *Denial of Service (D)*: the protection of the evidence collection and storage process from being attacked directly with the goal of disabling or shutting it down completely (e.g., to cover-up simultaneous attacks on another service). Possible adversaries are external attackers. This is a very generic threat that cannot be addressed by a single tool or control but rather requires a set a measures (on the network and application layer) to enhance denial of service resilience.
- *Evidence manipulation (T,R,I)*: the protection of evidence from intentional manipulation (e.g., deletion of records, changing of contents, manipulation of timestamps). Possible adversaries are malicious insiders and external attackers. Using Insynd for evidence collection and storage addresses this threat, since Insynd provides tampering and deletion detection.

Some of these threats can be mitigated by implementing appropriate security controls (i.e., using Insynd for evidence transport and storage). It provides effective protection by employing security techniques described in Section 3.

### 6.4 Requirements Evaluation

In this section, we evaluate the integration of Insynd against the requirements described in Section 4. In step 1 of the fictitious scenario, the data minimization principle is being followed because the specialized agent only collects evidence on the existence of snapshots.

This workflow is secure as soon as the collection agent inserts data into the evidence store in step 2. More precisely, evidence records are tamper-evident and encrypted. This is true, even though the evidence is actually stored on an untrusted virtual machine. The only way to compromise evidence now, is to attack the availability of the server hosting the Insynd server.

When the processing agent in step 3 retrieves records for evaluation, it can be assured of the authenticity of the data and that it has been provably collected by a collection agent. Since evidence records may be subject to maximum data retention regulation, records that are not needed anymore are deleted.

As previously mentioned in Section 5 we use JADE as an agent runtime. To secure our system against non-authorized agents, we use the TrustedAgents add-on for the JADE platform. This ensures that only validated agents are able to join our runtime environment. This effectively prevents agent injection attacks, where malicious agents could be inserted at either the collection or processing side to compromise our system.

As can be seen, the evidence records are protected all the way from the evidence source to the processing agent using only encrypted communication channels and having an additional layer of security (message encryption) provided by Insynd. Additionally, while the evidence is being stored, it remains encrypted.

### 6.5 Scalability

Obviously, since there is a vast amount of evidence sources and therefore a potentially equal number of collection agents, ensuring the scalability of the process and the implementation is very important. This has been considered very early in the design process by choosing an software agent-based approach for the system architecture. Software agents are inherently distributable and allow for complex message flow modeling in an infrastructure. Therefore, the core components evidence collection, storage and processing become distributable as well. In our future work, we'll focus on the scalability aspects. We will follow a methodology where we focus on the following technical key scalability indicators:

- Data transfer volume: amount of evidence data being transferred over the network
- Message volume: amount of evidence message transmissions over the network
- Storage volume: amount of storage required for evidence
- Encryption overhead: performance impact introduced by encryption and decryption

Based on the identified performance impact of each of these indicators, in the second step, we model different message flow optimization strategies to alleviate their impact and ensure scalability.

## 7 CONCLUSIONS

In this paper, we presented our system design and implementation for secure evidence collection in cloud computing. The evidence provides the general basis for performing cloud accountability audits. Accountability audits take a large variety of evidence sources and data processing requirements into account.

We showed what the requirements for a secure evidence collection process are and demonstrated how these issues are addressed by incorporating Insynd into our system. We described how the core principles of digital evidence are addressed by our system. Additionally, we considered data protection principles for the evidence collection process, how they influence our approach and how they are addressed in our system by integrating Insynd. For this, we presented the relevant architectural parts of our prototype.

In our future work, we will focus on the scalability of our audit system in general and the scalability of the components involved in evidence collection in particular. For that reason, we will focus on the distribution of the audit system and evidence collection not only in the same domain (i.e., in the same infrastructure), but also taking into account outsourcing and multi-provider collection scenarios.

## ACKNOWLEDGEMENTS

This work has been partly funded from the European Commissions Seventh Framework Programme (FP7/2007-2013), grant agreement 317550, Cloud Accountability Project - http://www.a4cloud.eu/ - (A4CLOUD).

# REFERENCES


An, J. H. (2001). Authenticated encryption in the public-key setting: Security notions and analyses. *IACR Cryptology ePrint Archive*, 2001:79.

Bellare, M. and Yee, B. (2003). Forward-security in private-key cryptography. In *Topics in Cryptology—CT-RSA 2003*, pages 1–18. Springer.

Bernstein, D. J., Lange, T., and Schwabe, P. (2012). The security impact of a new cryptographic library. In Hevia, A. and Neven, G., editors, *Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings*, volume 7533 of *Lecture Notes in Computer Science*, pages 159–176. Springer.

Bowers, K. D., Hart, C., Juels, A., and Triandopoulos, N. (2014). PillarBox: Combating Next-Generation Malware with Fast Forward-Secure Logging. In *Research in Attacks, Intrusions and Defenses Symposium*, volume 8688, pages 46–67. Springer.

Dingledine, R., Mathewson, N., and Syverson, P. F. (2004). Tor: The second-generation onion router. In Blaze, M., editor, *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320. USENIX.

Doelitzscher, F., Reich, C., Knahl, M., Passfall, A., and Clarke, N. (2012). An Agent Based Business Aware Incident Detection System for Cloud Environments. *Journal of Cloud Computing: Advances, Systems and Applications*, 1(1):9.

Doelitzscher, F., Ruebsamen, T., Karbe, T., Reich, C., and Clarke, N. (2013). Sun behind clouds - on automatic cloud security audits and a cloud audit policy language. *International Journal On Advances in Networks and Services*, 6(1 & 2).

Gupta, A. (2013). Privacy preserving efficient digital forensic investigation framework. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 387–392.

Haeberlen, A. (2009). A case for the accountable cloud. In *Proceedings of the 3rd ACM SIGOPS International Workshop on Large-Scale Distributed Systems and Middleware (LADIS'09)*.

JADE (2015). Java Agent DEvelopement framework. http://jade.tilab.com.

Jansen, W. and Grance, T. (2011). Sp 800-144. guidelines on security and privacy in public cloud computing. Technical report, Gaithersburg, MD, United States.

Lopez, J., Ruebsamen, T., and Westhoff, D. (2014). Privacy-friendly cloud audits with somewhat homomorphic and searchable encryption. In *Innovations for Community Services (I4CS), 2014 14th International Conference on*, pages 95–103.

Microsoft Developer Network (2015). The Stride Threat Model. https://msdn.microsoft.com/en-US/library/ee823878(v=cs.20).aspx.

Mohay, G. M., Anderson, A. M., Collie, B., de Vel, O., and McKemmish, R. D. (2003). *Computer and Intrusion Forensics*. Artech House, Boston, MA, USA. For more information about this book please refer to the publisher's website (see link) or contact the authors.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28.

OpenStack (2015). Openstack. http://www.openstack.org/.

Pearson, S. (2011). Toward accountability in the cloud. *Internet Computing, IEEE*, 15(4):64–69.

Pulls, T. and Peeters, R. (2015a). Balloon: A forward-secure append-only persistent authenticated data structure. Cryptology ePrint Archive, Report 2015/007.

Pulls, T. and Peeters, R. (2015b). Insynd: Secure one-way messaging through Balloons. Cryptology ePrint Archive, Report 2015/150.

Pulls, T., Peeters, R., and Wouters, K. (2013). Distributed privacy-preserving transparency logging. In Sadeghi, A.-R. and Foresti, S., editors, *WPES*, pages 83–94. ACM.

Redfield, C. M. and Date, H. (2014). Gringotts: Securing data for digital evidence. In *Security and Privacy Workshops (SPW), 2014 IEEE*, pages 10–17.

Ruebsamen, T. and Reich, C. (2013). Supporting cloud accountability by collecting evidence using audit agents. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 185–190.

Weitzner, D. J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., and Sussman, G. J. (2008). Information accountability. *Commun. ACM*, 51(6):82–87.

Zhang, R., Li, Z., Yang, Y., and Li, Z. (2013). An efficient massive evidence storage and retrieval scheme in encrypted database. In *Information and Network Security (ICINS 2013), 2013 International Conference on*, pages 1–6.