TOWARDS ENHANCING SECURITY IN CLOUD STORAGE ENVIRONMENTS

by

Duane C. Wilson

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

Feb, 2016

Abstract

Although widely adopted, one of the biggest concerns with cloud computing is how to preserve the security and privacy of client data being processed and/or stored in a cloud computing environment. When it comes to cloud data protection, the methods employed can be very similar to protecting data within a traditional data center. Authentication and identity, access control, encryption, secure deletion, integrity checking, and data masking are all data protection methods that have applicability in cloud computing. Current research in cloud data protection primarily falls into three main categories: 1) Authentication & Access Control, 2) Encryption, and 3) Intrusion Detection. This thesis examines the various mechanisms that currently exist to protect data being stored in a public cloud computing environment. It also looks at the methods employed to detect intrusions targeting cloud data when and if data protection mechanisms fail. In response to these findings, we present three primary contributions that focus on enhancing the overall security of user data residing in a hosted environment such as the cloud. We first provide an analysis of Cloud Storage vendors that shows how data can be exposed when shared - even in the most

ABSTRACT

'secure' environments. Secondly, we offer Pretty Good Privacy (PGP) as a method of securing data within this environment while enhancing PGP's Web of Trust validation mechanism using Bitcoin. Lastly, we provide a framework for protecting data exfiltration attempts in Software-as-a-Service (SaaS) Cloud Storage environments using Cyber Deception.

Primary Reader: Giuseppe Ateniese

Secondary Reader: William Agresti

Acknowledgments

I would like to thank all of the people who supported me through this process, especially my wife - Natasha Wilson - for the tireless hours she spent listening, encouraging, and keeping things running while I pursued my lifelong dream. I could not have done this without her support. I also want to thank God for Him giving me a heart of persistence throughout this arduos process. I want to thank my advisor Giuseppe Ateniese for his help and guidance throughout this process. Last, but not least, I would like to express my gratitude to those who have agreed to help improve my research outcomes through the review of this work.

Dedication

This thesis is dedicated to the people I am called to inspire and mentor as a result of this pursuit. Accomplishments of this nature are never for yourself, but for those who come after and are motivated by your example ...

x

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service, three service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), and four deployment models: Private Cloud, Community Cloud, Public Cloud and Hybrid Cloud.[1]

In SaaS, Cloud application services deliver software as a service over the Internet, eliminating the need to install and run the application on the customer's own computers and simplifying maintenance and support. In PaaS, Cloud platform services

Figure 1.1: NIST Cloud Computing Model

deliver a computing platform and/or solution stack as a service, often consuming cloud infrastructure and sustaining cloud applications. Lastly, in IaaS, Cloud infrastructure services, deliver computer infrastructure - typically a platform virtualization environment as a service, along with block storage and networking.

In a Private cloud, the cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers. It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises. A Community cloud is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community,

a third party, or some combination of them, and it may exist on or off premises. A Public Cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider. Lastly, in a Hybrid cloud, the cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).[1]

1.1 Background

Although widely adopted, one of the biggest concerns with cloud computing is how to preserve the **security** and **privacy** of client data being processed and/or stored in a cloud computing environment. These concerns primarily exist because the traditional system boundaries are not present in the cloud and multiple tenants will commonly share the cloud infrastructure as is the case in a Public Cloud offering. Multi-tenancy refers to a principle in software architecture where a single instance of the application runs on a server, serving multiple tenants. This is in contrast to a multi-instance architecture where separate software instances (or hardware systems) are set up for different client organizations. This issue is further complicated when

more than one cloud service provider is involved in a user/provider interaction.

There are a number of methods that have been proposed to handle the concerns of security and privacy in cloud computing, one of which is cloud data protection. When it comes to cloud data protection, methods can be similar to protecting data within a traditional data center. Authentication and identity, access control, encryption, secure deletion, integrity checking, and data masking are all data protection methods that have applicability in cloud computing. **Authentication** of users takes several forms, but all are based on a combination of *authentication factors*: something an individual knows (such as a password), something they possess (such as a security token), or some measurable quality that is intrinsic to them (such as a fingerprint). When we discuss **Access Controls**, we refer to: 1) Subjects which are people or processes acting on their behalf or 2) Objects such as files or other resources (a directory, device, or service of some sort). Access controls are generally described as either discretionary or non-discretionary, and the most common access control models are: Discretionary Access Control, Role-based Access Control, and Mandatory Access Control. **Encryption** is a key component to protect data at rest in the cloud. There are multiple ways of encrypting data at rest in the cloud. These include: full disk, directory level (or file system), file level, or application level. As usual, critical to implementing any of these forms of encryption is the need to manage the keys that are used to encrypt and decrypt data. **Integrity Checking** is typically performed via a hash function or similar construct (e.g. Cyclic Redundancy Check or CRC)

that is computed at the time a file is created in the cloud and subsequently saved. The check is performed by recomputing the hash function output or CRC prior to editing the file to verify that the original value matches. Clearing and Sanitization are methods most commonly employed when **Deleting** cloud data. The *DoD 5220.22-M, National Industrial Security Program Operating Manual* describes these methods in more detail. It is important to note that data stored in a public cloud is not often sanitized to DoD levels. Lastly, **Data Masking** is a technique that is intended to remove all identifiable and distinguishing characteristics from data in order to render it anonymous and yet still be operable.[2]

In the event that the aforementioned cloud data protection mechanisms fail or are compromised, it is important to be able to accurately detect the issue. Although distributed Intrusion Detection System (IDS) technology has been tested to be capable of working well in some large scale networks, the utilization and deployment in Cloud Computing is still a challenging task. The complex architecture of a Cloud infrastructure and the different kinds of users lead to different requirements and possibilities for being secured by IDS. An important issue for the user is not having full control over the used infrastructure. Although some of the Cloud users require to separate the IDS and the monitored target, it is still necessary for the Cloud provider to provide such a possibility that certain end users can fully control at least the currently used resources.[3] To counter a number of these aforementioned challenges, in chapters 3, 4, and 5 we look at ways to securely share data in the cloud, offer better security

guarantees to users in terms of data protection, and to mitigate data theft in a cloud storage environment. In the next chapter, we expound upon the prior work that seeks to address data protection in a cloud storage environment.

Chapter 2

Prior Work

As mentioned in chapter 1, three of the ways in which data can be protected in the cloud are: Strong Access Control and Authentication, Data Encryption, and Intrusion Detection. In this chapter, we expound upon these areas and detail prior work that focuses on the danger of relying upon these methods alone for data protection. This sets a precedent for our research contributions to follow. In chapters 3, 4, and 5 we present related work specific to our contributions: Data Sharing in the Cloud, Enhanced PGP using Bitcoin, and the Mitigation of Data Exfiltration in SaaS cloud environments.

2.1 Overview

Maintaining confidentiality, integrity, and availability for data security is a function of the correct application and configuration of familiar network, system, and application security mechanisms at various levels in the cloud infrastructure. Among these mechanisms are a broad range of components that implement authentication and access control. Authentication of users and even of communication systems is performed by various means. Single factor authentication is based on only one authentication factor. Stronger authentication requires additional factors; for instance, two factor authentication is based on two authentication factors (such as a pin and a fingerprint). Authentication is usually predicated on an underlying identity infrastructure. The most basic scheme is where account information for one or a small number of users is kept in flat files that are used to verify identity and passwords, but this scheme does not scale to more than a very few systems.

The key to effective access controls is the centralization of identity. One problem with using traditional identity approaches in a cloud environment is faced when the enterprise uses multiple Cloud Service Providers. In such a use case, synchronizing identity information with the enterprise is not scalable. These and other issues arise when migrating infrastructure toward a cloud-based solution. Infrastructure tends to employ domain-centric identity approaches that do not allow for looser alignment such as with partnership. For these reasons, federated identity management (FIM) is an effective foundation for identity in cloud computing.

Authentication and access control serve as the first layer of the Defense in Depth posture within a Cloud Storage Environment. Intrusion Detection Systems (IDS) have been used widely to detect malicious behaviors in network communication and hosts in the event of an authentication or access control failure. IDS management is an important capability for distributed IDS solutions, which makes it possible to integrate and handle different types of sensors or collect and synthesize alerts generated from multiple hosts located in the distributed environment. Facing new application scenarios in Cloud Computing, the IDS approaches yield several problems since the operator of the IDS should be the user, not the administrator of the Cloud infrastructure. Extensibility, efficient management, and compatibility to virtualization-based context need to be introduced into many existing IDS implementations. Additionally, the Cloud providers need to enable possibilities to deploy and configure IDS for the user.

To overcome the threat of data compromise in the cloud, encryption is often heralded as be all end all solution. According to the Enterprise Cloud Computing Blog[4] three of the common security questions that should be asked when considering a cloud service provider for data storage are:

1. **Data on Wire.** Are files transferred to/from cloud servers encrypted?

2. **Data at Rest.** Are files stored on cloud servers encrypted?

3. **Data Retention.** If files on cloud servers are encrypted and there is a re-

quest from law enforcement to decrypt the data, then what do you do? Bonus question: What if you have the key(s)?

There are multiple ways of encrypting data at rest in the cloud. Full disk encryption is encryption of data at the disk level. With this method of encryption, the operating system, its applications, and the data residing on it are all encrypted simply by existing on the encrypted disk. The major concerns with this approach are that of performance and reliability. For example, even minor disk corruption can be fatal as the OS, applications, and data rely on the encryption infrastructure at this level. With file system encryption, entire data directories are encrypted or decrypted as a container. Access to files requires use of encryption/decryption keys. This approach can also be used to segregate data of identical sensitivity or categorization into directories that are individually encrypted with different keys. File level encryption is simply the use of encryption at the file level vs. the file system or disk. This is commonly the most efficient form of encryption. At the application level, encryption and decryption of data is managed by the applications themselves. In addition to the common key management challenges associated with encryption, identifying data recovery methods in the event that an encryption key is lost or inaccessible is critical.

2.2 Encryption in the Cloud

Even with its shortfalls, our research indicates that encryption is still the preferred method of protecting data in the public cloud. In lieu of this, this section primarily focuses on prior work that attempts to use encryption for various means of data protection in the cloud.

2.2.1 Searchable Encryption in the Cloud

In a typical searchable encryption scheme, only a single-user (i.e. only the holder of a secret key, which is referred to as query key) can issue valid search queries upon the cloud database. Yangs work considers the fact that there are cases in which searchable encryption needs to work in a multi-user setting (e.g. an enterprise outsources its database to the cloud, and authorizes multiple users).[5] He proposes an efficient multi-user searchable encryption scheme, which possesses the following beneficial features.

- **Distinct Query Keys**. Each authorized user has a distinct query key for constructing search queries. This makes user revocation and accountability possible.

- **Complete User Revocation.** Allows for very efficient user revocation: revocation of a user does not affect other non-revoked users at all, requiring neither key renewal for non-revoked users, nor update to the encrypted database includ-

ing the index. Moreover, revoked users completely lose their search privileges, given that the semi-trusted cloud destroyed the related helper keys which aid in processing queries.

- **Exculpability**. Scheme achieves exculpability which ensures that no one (including the cloud) can generate valid search queries on behalf of a user. Exculpability turns out to be an important property in the multi-user setting where accountability is desired.

In their work, Wang et. al. consider the large number of data users and huge amount of outsourced data files in the cloud. This problem is particularly challenging as it is extremely difficult to meet also the practical requirements of performance, system usability, and high-level user searching experiences. Their work investigates these challenges and defines the problem of fuzzy keyword search over encrypted cloud data, which should be explored for effective data utilization in Cloud Computing. Fuzzy keyword search aims at accommodating various typos and representation inconsistencies in different user searching input for acceptable system usability and overall user searching experience, while protecting keyword privacy. The overall service design will provide built-in security assurances of keyword privacy and data file confidentiality at an acceptable cost and benefit cloud customers who seek to utilize the cloud with strong privacy guarantee. In addition, it is also expected to enable cloud service providers to securely and effectively deliver value from the cloud infrastructure to their enterprise customers, significantly encouraging the adoption of the

cloud.[6]

Prior works on query processing on encrypted data did not provide data confidentiality guarantees when data is stored and accessed in the cloud. Tradeoffs between secrecy and efficiency needs to be made when satisfying both aspects of data confidentiality while being suitable for practical use. Second, to support common relational data management functions, various types of queries such as exact queries, range queries, data updates, insertion and deletion should be supported. In their work, Wang e.t. al. proposes a comprehensive framework for secure and efficient query processing of relational data in the cloud. Their framework ensures data confidentiality using a salted Information Dispersal Algorithm encoding scheme and column-access-via-proxy query processing primitives, and ensures query efficiency using matrix column accesses and a secure B+-tree index. In addition, they also make claims of data availability and integrity.[7]

Koletka and Hutchison discuss a solution to cloud security and privacy that allows users to securely store data on a public cloud, while also allowing for searchability through the users encrypted data. Users are able to submit encrypted keyword queries and, through a symmetric searchable encryption scheme, the system finds all files with such keywords contained within. The system is designed in such a manner that trust from a public cloud provider is not required. The solution satisfies data confidentiality and integrity, file sharing is catered for, and a user key-revocation scheme is in place. Encryption provides security in the event of a data breach.[8]

Current encrypted search schemes that allow multiple users to read and write to the database have two major drawbacks: (i) they support only keyword searches or conjunctions of keywords, and (ii) do not allow the specifying of different access policies for different users. Instead, they assume all users have the same rights. Ion e.t. al. show an implementation of a novel scheme that addresses both issues. The presented scheme allows multi-users to perform complex, SQLlike queries on the encrypted database without revealing the query to the cloud server, and allows enforcing different access control policies for the users, in a single, integrated solution. Lastly, the scheme does not leak to the cloud provider information on the access policies.[9]

2.2.2 Secure Information Sharing for Cloud

In[10] Li et. al. propose a mechanism in which the cloud platform is utilized to accomplish electronic medical record exchange, and at the same time used to protect the patients privacy. The electronic medical record numbers, generated using the SID in the health data card, random value, and treatment serial number, are all different even for the same patient for each of his or her electronic medical record numbers. It has the unlinkability characteristic. In addition, a one time key is used for medical record encryption to enhance the security of the encrypted section. The utilization of the cloud platform not only allows hospitals with fewer resources to receive the electronic medical record service and save electronic medical records, but also allows

patients to review their own medical records at home. In the future, cloud technology may be used to review medical records on mobile devices, instead of the service being limited to a sole terminal device such as a computer.

One way to handle the issues of security and privacy as it relates to cloud data storage is to support data replication and distribution on the cloud via a local, centrally synchronized storage. In[11] Pagano proposes the use of an in-memory RDBMS with row-level data encryption for granting and revoking access rights to distributed data. It can be used in the cloud to manage very granular access rights in a highly distributed database. This allows for stronger confidence in the privacy of **shared** sensitive data. An interesting field of application is the use in (business) cooperative environments (e.g. professional networks). In these environments, privacy is a priority, but low computing resources don't allow the use of slow and complex algorithms. IMDBs and their smart encryption technique achieve the goal in a more effective way.

In[12] Aniello, et. al. describe the architecture of a basic secure file sharing facility relying on a multi-party threshold-based key-sharing scheme that can be overlaid on top of the existing stackable networked file systems. They discuss its application to the implementation of distributed cryptographic file systems. They focus on the following 3 properties concerning a shared file on a group of n users with a (secret sharing) threshold k n: 1) the file creator only needs to know the public keys of all the group members, 2) whoever wants to access (read/write) the shared file, must be a group member and must contact at least k 1 group members, 3) no user outside

the group can gain access to a shared file even if he receives help from all members of the group.

Sohn et. al.[13] propose a method for sharing User Generate Content (UGC) securely based on the personal information of users. With the proposed method, virtual secure space is created for content delivery. The virtual secure space allows UGC creator to deliver contents to users who have similar personal information and they can consume the contents without any leakage of personal information. This scheme restricts information access to specific groups of individuals with similar interests and profiles by creating encrypted vaults accessible by users with similarities (e.g. age, hobbies, occupation, etc). The advantage of this approach is that creators would not have to be concerned about potential misuse of their content by being assured that their content are delivered to users with similar preferences. This will also aid in organizing content.

The same group of authors[14] present the design of a secure cache system that allows roaming users to cache files on untrusted file hosting servers - like the cloud. The system allows flexible sharing of cached files among unauthenticated users and does not require a global authentication framework. Files are encrypted when they are transferred over network and stored and system uses Public Key cryptography which enables secure information sharing in the untrusted environment. Sharing is discussed as follows: when a roaming user wishes to give another user read access to one of his/her files, the decryption key is provided to that user - along with information

required to access the file on cache server. It is also possible to share encryption keys with other users - at a user's discretion.

In[15] Yen-Hung et. al. propose and develops a scalable Secure Enterprise File Sync and Share (EFSS) service which can be deployed on the OpenStack cloud infrastructure to securely provide employees with EFSS service. Security issues, including employee privacy protection, management of share links and synchronized cloud files, and the secure enterprise directory integration, are discussed in this article. Secure sharing is accomplished in their scheme via Distinct Share Links - which is able to protect the share link by decreasing its diffusibility and adding traceability and controllability to it. To protect a share link, a distinct share link layer is created between the share link and clients, and the Distinct Share Links in the additional layer are pointing to the share link. Moreover, the Distinct Share Links are attached with permissions, identities, and access conditions, they are then sent to different recipients according to their identities. If the recipient passes the identity and access condition checks, the Distinct Share Link then prepares an ephemeral representation for the recipient to access the share link.

Secure and trustworthy file sharing over cloud storage using eID tokens[16] presents a multi-platform, open-source application that aims to protect data stored and shared in existing cloud storage services. The access to the cryptographic material used to protect data is implemented using the identification and authentication functionalities of national electronic identity (eID) tokens. The solution provides the following

benefits: (i) confidentiality, to prevent non-authorized readings, (ii) integrity control, to detect malicious tampering, (iii) protection against unwanted file removals, either by malicious or legitimate persons, and (iv) access control to the shared data based on strong identification and authentication of people, using the nowadays widespread electronic, personal identity tokens (eIDs for short). Identification and authentication of users in Protbox was accomplished using national eID tokens - since they already contained X.509 authentication certificates and embedded Public Keys to validate signatures of Pair Key requests and responses. In this way, the access to protected files shared through the cloud only occurs after a two-factor authentication takes place.

Lastly, Shu et. al. present Shield[17] which is a stackable secure storage system for file sharing in public storage. Shield is designed to secure data storing and data sharing inside the trust domain (e.g., an organization or department) under the network and storage environments shared among multiple parties, and to save file owners from tedious management without fully trusting the cloud server. Design goals and benefits are: Underlying file systems independence, End-to-end protection for confidentiality and integrity, Keys management and key distribution, Efficient permission revocation and concurrent writing support. Shield excludes the secret keys management from the responsibility of the cloud server to minimize the risk of user data compromise by the provider themselves or a malicious adversary. The cloud server is only responsible for storing the files and providing access control for the cipher text

(restricting data access to authorized users). The Proxy Server (PS) in their model can serve as a file owner defined delegate as long as it is in the same trust domain - serving as an independent third party. It is primarily responsible for processing users' access requests by distributing the corresponding secret keys according to their access permissions.

2.2.3 Secure Cloud Storage

Data confidentiality is one of the key concerns that prevent organizations from widely adopting third-party computing clouds. Puttaswamy et. al. describe a set of techniques that promote data confidentiality on the cloud using end-to-end data encryption. Encrypted data on the cloud prevents privacy leakage to compromised or malicious clouds, while users can easily access data by decrypting data locally with keys from a trusted organization. Using dynamic program analysis techniques, they have been able to automatically identify functionally encryptable application data, data that can be safely encrypted without negatively affecting application functionality. Through the modification of the application runtime engine, they show how an optimal assignment of encryption keys can be determined that minimizes key management overhead and impact of key compromise. The major benefit of their approach is that applications running on the cloud can protect their data from security breaches or compromises in the cloud through the adoption of these methods.[18]

In their work[19] Bessani et. al. present a dependable and secure storage system

that leverages the benefits of cloud computing by using a combination of diverse commercial clouds to build a cloud-of-clouds. In other words, their implementation is a virtual storage cloud, which is accessed by its users by invoking operations in several individual clouds. It addresses four important limitations of cloud computing for data storage in the following way:

- **Loss of availability:** approach handles this problem by exploiting replication and diversity to store the data on several clouds, thus allowing access to the data as long as a subset of them is reachable.

- **Loss and corruption of data:** approach handles problem using Byzantine fault-tolerant replication to store data on several cloud services, allowing data to be retrieved correctly even if some of the clouds corrupt or lose data.

- **Loss of privacy:** approach employs a secret sharing scheme and erasure codes to avoid storing clear data in the clouds and to improve the storage efficiency, amortizing the replication factor on the cost of the solution.

- **Vendor lock-in:** approach does not depend on a single cloud provider, but on a few, so data access can be balanced among the providers considering their practices (e.g., what they charge). Second, it uses erasure codes to store only a fraction (typically half) of the total amount of data in each cloud. In case the need of exchanging one provider by another arises, the cost of migrating the data will be at most a fraction of what it would be otherwise.

In,[20] Huang et. al. propose a privacy-preserving cloud storage framework, which includes the design of data organization structure, the generation and management of keys, the treatment of users access right changes, dynamic operations of data, and the interaction between participants. They have designed an interactive protocol and an extirpation-based key derivation algorithm, which are combined with lazy revocation, multi-tree structure and symmetric encryption to form a privacy-preserving, efficient framework for cloud storage. Lastly, they present their analysis of the effectiveness of extirpation-based key derivation, system overhead, and validate their privacy-preserving claims.

SiRiUS[21] can work over insecure file systems as a cryptographic storage layer to supply storage security. Plutus[22] offers cryptographic group sharing with lazy revocation, random access, and file name encryption. CRUST[23] is a stackable secure file system with completely symmetrical encryption and in-band key distribution. However, Plutus, SiRiUS and CRUST all require the file owner to bear the burden of dominating the access control and key distribution (although there are some differences in key distribution: Plutus needs users to get the file key from the file owner, while SiRiUS and CRUST require pre-sharing of some message of key materials among users before file access). Moreover, SiRiUS employs aggressive revocation which requires far more intensive computations than lazy revocation and CRUST relies on some global shared data structures to distribute keys and to retrieve previous states for revocation. Sharing models such as these allow the users to have control over the

sharing of their files as well as distributing the keys to access the files of interest.

Companies like Navajo Systems[24] and Ciphercloud[25] provide a trusted application-level proxy that intercepts network traffic between clients and cloud-hosted servers (e.g. IMAP), and encrypts sensitive data stored on the server. These products appear to breakup sensitive data (specified by application-specific rules) into tokens(such as words in a string), and encrypt each of these tokens using an order-preserving encryption scheme, which allows token-level searching and sorting. SUNDR[26] uses cryptography to provide privacy and integrity in a file system on top of an untrusted file server. Using a SUNDR-like model, SPORC[27] and Depot[28] show how to build low-latency applications, running mostly on the clients, without having to trust a server. However, existing server-side applications that involve separate database and application servers cannot be used with these systems unless they are rewritten as distributed client-side applications to work with SPORC or Depot. Many applications are not amenable to such a structure.

Lastly, the following cloud storage security solutions, implemented as third-party software applications, possess similar features to the Secure CSPs we analyzed: BoxCryptor,[29] Viivo,[30] CloudFogger,[31] Sookasa,[32] TrueCrypt[33] and CCE (Citizen Card Encrypted).[34] For the majority of these solutions, encryption keys and the sharing logic of these is handled within a backend platform available in a web server. Users must implicitly trust that the web server is safeguarded. For example, in BoxCryptor, file sharing is targeted to individual files, where a random key is generated to encrypt

every different file that can be shared with another single user or with a group of users. This key is then stored in the BoxCryptor Key Server and made accessible to the intended user or group. As security measures, these keys are encrypted with cryptographic material generated from the users credentials and stored locally. The remaining solutions allow the sharing of whole directories with specific users, generating an encryption key per directory and storing it in the applications backend servers, with access limited to those users. The encryption material relevant to file protection is said to be kept locally, without ever being transferred to these backend servers.

For a more controlled sharing protocol, Viivo proposes a mediator-based implementation where every shared folder has a user with moderator privileges, which, by default, is the first user to attain access to said folder. New users must request for permission of access to the encrypted contents directly to the moderator, and this moderator must constantly check for and manage these requests. Since there is only one moderator per folder, this moderator must be familiar with all of the requesting users. With this, a user that is only known as trustworthy by a single or a few users of the shared folder excluding the moderator will, more likely than not, have his request denied. With Protbox, every single sharing request is sent in a multicast fashion, and the requirements for one of these requests to be accepted is to provide a valid certificate chain and a valid signature and to have at least one user accept such a request. This solution does not have a central authority controlling ownership rights over Shared Folders; everyone that has access to the Shared Folder is a peer

with equal rights. These approaches all discuss ways in which secure sharing could be approached in a hosted environment such as the cloud. As stated above, in most of the solutions users must implicitly trust that the web server is safeguarded. In the next chapter we provide the results of our analysis of cloud vendors claiming full security within their environments. We are able to demonstrate through our analysis that these claims are only valid if data is not shared among cloud users.

Chapter 3

Dangers of Sharing in Secure Cloud Environments

With the advent of cloud computing, a number of Cloud Storage Providers (CSPs) have arisen to provide Storage-as-a-Service (SaaS) offerings to both regular consumers and business organizations. SaaS refers to an architectural model in which a cloud provider provides digital storage on their own infrastructure.[36] Three models exist amongst SaaS providers for protecting the confidentiality of data stored in the cloud: 1) no encryption (data is stored in plain text), 2) server-side encryption (data is encrypted once uploaded), and 3) client-side encryption (data is encrypted prior to upload). Dropbox is the most popular version of a cloud storage provider that adheres to the first confidentiality model. In this chapter we examine secure alternatives to Dropbox that provide client-side encryption. Our primary motivation is based on

consistent claims made by CSPs that guarantee the confidentiality of data stored in the cloud. The major claims are as follows:

- "No one unauthorized not even the cloud storage provider can access the files."[37]
- "Our 'zero-knowledge' privacy environment ensures we can never see your data. Not our staff. Not the government. Not anyone."[38]
- "Contrary to other solutions, no storage provider or network administrator, no unauthorized hacker, not even we can read your files."[39]

These principles of confidentiality hold true in use cases where data is not shared with other cloud users or with entities outside of the cloud storage environment (e.g., non-members). In the evaluated SaaS environments, data sharing is accomplished in three ways: Web Link, Folder, or Group, the latter two being the focus of this research. Through our analysis we discovered that for each data sharing mechanism, there are inherent weaknesses that can expose user data to the cloud provider which directly contradicts the aforementioned CSP claims. What would prompt a Cloud Provider to compromise the trust of its users in this way?

1. **National Security**: In the interest of National Security, governments will often collect citizen data for the purposes of confirming a threat. This collection typically consists of activities such as wire tapping, data harvesting, and other forms of information collection. Data Harvesting (usually associated with

Cloud-Storage attacks) refers to the collection of disparate data from a homogeneuous location. This approach is advantageous for a Government seeking data points from multiple entities such as is the case in a co-resident Cloud Storage environment.

2. **Oppressive Government**: Under certain government regimes, there may exist situations in which a Government might "force" a Cloud Provider to comply with new or existing disclosure regulations. Reasons for this may include: 1) Company Sanctions, 2) Periodic Evaluations, or 3) to Limit Monopoly business practices.

3. **Data Leakage Confirmation** As is common in the "Networked" world, data that is accessible via the Internet is subject to data breaches. The Data Breach Notification Laws require states to report the occurrence of company and state-related data breaches. In order to confirm and subsequently comply with these laws, Cloud Providers could potentially need to access user data in plain text form.

We discuss the following contributions in this chapter:

- We present an overview of the various sharing scenarios employed by each evaluated CSP

- We highlight the weaknesses found in each CSP sharing scenario. Our research

focus is on exposing the weaknesses in private Group and Folder sharing scenarios.

- We describe how an attack against private Group or Folder sharing functions could work in practice.
- We provide evidence of Certificate Authority functionality via network traffic and source code analysis.
- We reverse engineer various CSPs code to reveal evidence that substantiates our claims that user data is not 100% safe from being read and/or manipulated by the CSP.
- We provide suggestions for addressing the inherent weaknesses discovered in the design of the CSP sharing functions.

3.1 Related Work

This section puts our analysis in perspective by examining preceding work that relates to the analysis of Cloud Storage Providers (CSPs). Prior work falls into three specific categories: 1) General Analysis of CSP security capabilities, 2) Analysis of the Design and Implementation of CSPs, and 3) Analysis of the Weaknesses associated with CSPs.

In,[40] Borgmann and Waidner of the Fraunhofer Institute for Secure Information

Technology, they studied the security mechanisms of seven Cloud Storage Services: CloudMe, CrashPlan, Dropbox, Mozy, TeamDrive, Ubuntu One, and Wuala. The study focused on the following security requirements: Registration and Login, Transport Security, Encryption, Secure File Sharing, and Secure Deduplication. Similar to our approach, they examine several aspects of the CSP's file sharing mechanisms for security flaws. Specifically, they highlight the fact that if client-side encryption is used, sharing should not weaken the security level. In particular, the CSP should not be able to read the shared files. The scope of their work covered sharing with other subscribers of the same service, sharing files with a closed group of non-subscribers, and sharing files with everybody. As we also discovered in our research, sharing via secret web link (e.g., closed group of non-subscribers) or making data public reveals shared data to the CSP. Their analysis, however, did not yield the fact that sharing files with subscribers could also result in a breach of confidentiality with the CSP due to the CSP acting in a Certificate Authority (CA) capacity. This is proven in our analysis and can serve as a viable extension to their work.

Mager et al. in[41] examine the design and implementation of an online backup and file sharing system called Wuala. The goals of their work consist of four primary items: Characterization of the Infrastructure, Understanding the Data Placement Methodology, Identifying the Coding Techniques relating to Data Availability (accessibility of files at any time) and Durability (ensuring files are never lost), and the Determination of the Data Transport Protocol used. The scope of their evaluation

does not necessarily include security considerations, although the data structure employed by Wuala for sharing data is mentioned as well as the type of encryption used for performing client-side encryption. Overall, this research provided useful insight into methods employed by the CSP to facilitate client usage of their infrastructure, however, it does not examine the infrastructure or design in detail for security weaknesses. As it pertains to their goals, our work differs in that it looks at the CSP infrastructure security features for consistency with user expectations. We focus on the data sharing (not placement) methodology employed by the CSPs. The coding techniques we identified relate to the discovery of evidence that the CSP is operating in a CA capacity during sharing transactions.

Finally in,[42] Kholia and Wegrzyn analyze the Dropbox cloud-based file storage service from a security perspective. Their research presents novel techniques to reverse engineer frozen Python applications (to include Dropbox). They specifically describe a method to bypass Dropbox's two factor authentication and hijack Dropbox accounts. Additionally, they introduce generic approaches to intercept SSL data using code injection techniques and monkey patching. This work is consistent with our analysis of CSPs for major security flaws as it results in the exposure of private user data and enables the CSP to have access to shared user data.

3.2 Cloud Storage Provider Overview

This section provides an overview of the Cloud Storage Providers (CSPs) we evaluated and the 'confidentiality model' they each employ to protect the confidentiality of user data. We define a confidentiality model as the method employed by the CSP to protect the confidentiality of users' stored data. According to,[43,44] there are several CSPs that offer client-side encryption. For our analysis, we focused on Wuala, Spider Oak, and Tresorit. Wuala encryption is performed with AES256 prior to files being uploaded. Encryption comprehensively includes not only file content, but also: file names, preview images, folders and metadata. An RSA2048 key is used for signatures and key exchange when folders are shared while SHA-256 is used for data integrity. Spider Oak can be used to share and back up files. Data is encrypted on a user computer with AES256 in CFB mode and HMAC-SHA256. As mentioned previously, the company claims to have no knowledge of what data is stored in their servers or user passwords. Their software works in smart phones, the Linux operating system, and Windows.[43] Lastly, Tresorit is a Hungarian-based company that uses AES256 to encrypt data before uploading it to the cloud. The company is offering $10,000.00 (US) to anyone who can break their security software. Similar to Spider Oak, data can be accessed via smart phone or desktop computer. Each CSP's confidentiality model is discussed in the following subsections. All information in this section has been adapted from the CSP's website and/or documentation unless otherwise specified.

Table 1 summarizes the confidentiality model employed by each CSP based on

	Sharing Scenario					
CSP	Public Web Link	Private Web Link	Public Folder	Private Folder	Public Group	Private Group
Wuala	Public URL	Private URL	Public Folder	Public Key Encryption	Public Group	Public Key Encryption
Spider Oak	Public URL	Public URL with Passwd	N/A	N/A	N/A	N/A
Tresorit	N/A	Encrypted Link	N/A	RSA or TGDH	N/A	ICE Protocol
Difficulty	Trivial	Trivial	Trivial	**Non-Trivial**	Trivial	**Non-Trivial**

Table 3.1: Trivial vs. Non-Trivial Sharing Scenarios

each data sharing scenario provided. It also presents an overview of each sharing scenario according to the level of effort required to 'exploit' the scenario (Trivial or Non-Trivial). Our research focuses on the sharing scenarios that the CSPs highlight as offering 100% data confidentiality (i.e., Private Group and Private Folder sharing). "N/A" represents a feature that is not implemented or has no documentation to support the scenario.

As noted above for all CSPs, when data is shared via Private web link, the CSP requires the transmission of some sensitive data in order to process the user's request. Similarly, each of the Public Sharing Scenarios (i.e., Web Link, Group, Folder) require implicit trust of the CSP in order to make the data public. As a result, these scenarios were discussed in this section for completeness but fall outside the scope of

our research due to this requirement of trust.

3.3 Threat Model

In this section we provide a threat model which describes how an attack against a Cloud Storage Provider (CSP), offering client-side encryption, could potentially be executed. Rating the threats we identify in this section are outside the scope of our research. We make the following assumptions with regards to the threat model:

1. The CSP client is trusted and has not been previously modified by a malicious insider or outsider. In the case of a modified client, users could easily be redirected to a rogue server upon client startup and be forced to perform all sharing transactions through this server.

2. The CSP server is trusted and has not been compromised. Similar to the case of a rogue client, a rogue server would allow an adversary access to user certificates and gain access to decryption keys.

3. Other CSP members can be trusted. When searching for an individual user in the CSP database, it is assumed that the users identified are legitimately who they appear to be.

4. The User certificates issued cannot be trusted because they are issued by the CSP.

5. Public or Private Web Link Sharing scenarios (as mentioned above) require users to reveal some form of sensitive information to the CSPs to enable decryption in a browser environment. Similarly, Public Groups and Folders are accessible by all members and pose no threats to users.

Our analysis consists primarily of network traffic analysis and reverse-engineering the source code of each CSP client through decompilation or disassembly. We selected this course of analysis due to fact that most of the CSP client code used some form of code obfuscation — making the replication of a rogue client challenging. Additionally, we could not produce counterfeit certificates 'on behalf of' other users, because we do not possess the signing key of the CSPs we analyzed. This signing key would be required to produce certificates that CSP client applications would trust. We discuss below a scenario that would enable a Cloud Provider to maliciously access a user's data.

1. User A signs up for Cloud Account
2. User A initiates sharing request with User B
3. Cloud client returns User B's contact information to User A
4. Cloud Provider substitutes its Public Key for User B
5. Without User A's knowledge his/her data is encrypted with the Public Key of the Cloud Provider

6. Cloud Provider is able to decrypt User A's data and view its contents

7. Cloud Provider then re-encrypts data with User B's Public Key and Cloud Client sends sharing request to User B

8. User B decrypts the data sent by User A w/o knowledge of the above-stated attack.

This scenario simply shows that because the CSPs are operating as Certificate Authorities, certificate manipulation, spoofing, or substitution of any kind can be accomplished. It is also important to emphasize that a malicious CSP cannot be detected since it can perform a standard *man-in-the-middle* attack in which the encrypted information flow from the sender is first decrypted and then re-encrypted on the fly under the recipient's public key (and vice-versa). In essence, the CSP is able to spoof the identity of any user through the use of certificates. In the next section, we show evidence to support our claim that each evaluated CSP is serving in a CA capacity - making all users susceptible to the aforementioned attacks.

3.4 Analysis Methodology and Results

In this section we provide a detailed review of our analysis results. It is important to note that although our results do show that each Cloud Storage Provider is in some capacity operating as a Certificate Authority (i.e., could issue fake certificates

to its users) - we did not find any evidence to support any claims of misuse by the Providers themselves (i.e., they are not actively exploiting users via this attack). Our analysis examined the sharing functions employed by each CSP which consisted of three phases: 1) Network Traffic Analysis (using Wireshark[46] and Fiddler Web Proxy Debugger[47]), 2) Source Code Decompilation (using AndroChef Java decompiler[48] and Boomerang C++ decompiler), and 3) Source Code Disassembly (using HopperApp[49] and Synalyze[50]). Evidence of CA functionality validates our assertion that when data is shared, it is possible for the CSP to view user data which contradicts their 100% data confidentially claims. Evidence of CA functionality includes (but is not limited to the following): Certificates (i.e., Root or Intermediary), Certificate Issuance, Certificate Validation, Certificate Revocation, Certificate Authentication, Certificate Renewal, Certificate Registration, Obtaining Certificates, Encryption/Decryption, and Digital Signature (signing or verification). The processes for performing the network traffic, source code decompilation, and source code disassembly analyses are discussed below.

A CA is a Trusted Third Party (TTP) organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The role of the CA in this process is to guarantee that the individual granted the unique certificate is, in fact, who he or she claims to be. CAs are a critical component in data security and electronic commerce because they guarantee that the two parties exchanging information are really who they claim to be.[51] Serving as a CA, the

CSPs would potentially be able to issue counterfeit certificates to users – pretending to be a legitimate entity that data is shared with. The code we identified falls into five categories: 1) Code Libraries, Certificate Files, Certificate Operations, Keystore Files/Definitions, and Peripheral Cryptographic Functions. Lastly, we focused on disassembling code that could not be reproduced in its original form through Decompilation. The code samples we extracted were only those that could potentially represent CA functionality. We primarily analyzed the main binary files (*.app*, *.exe* files) associated with each CSP.

3.4.1 Wuala Analysis

We started our analysis of Wuala by examining the network traffic that was generated between the client and server during sharing transactions and extracting any evidence of CA functionality. Through our analysis, we discovered evidence of a Wuala CA certificate which was used to confirm the validity of the other certificates below it in the certificate chain. A root certificate is either an unsigned public key certificate or a self-signed certificate that identifies the Root Certificate Authority (CA). It is the top-most certificate of the tree, the private key of which is used to "sign" other certificates. All certificates immediately below the root certificate inherit the trustworthiness of the root certificate.[52]

Secondly, during disassembly of the Wuala binaries, we identified multiple references to a 'WualaCACerts' file which contains several certificates that the Wuala

client uses during its execution to include the Wuala CA file — mentioned previously. The reference below is to the bouncy castle (Java Crypto Library) implementation of a Keystore which also contains individual certificate entries. The contents of this particular file is discussed in the decompilation section below. Lastly, during the decompilation phase, we discovered that Wuala uses a combination of Name Obfuscation (e.g., changing method and variable names while maintaining the functionality). To overcome this, we performed searches across the entire codebase for files that fit into the aforementioned categories. The primary results of the Wuala decompilation process yielded the WualaCACerts file. This file contains 4 certificate entries: *wualaadminca*, *wualaserverca*, *wualaca*, and *wualaclientca*. Each certificate is named to denote the function it serves within the operation of the Wuala application (i.e., administrative, client, server, and CA). Each certificate entry has an 'Extension' section with an ObjectID and a specification of what the key (within the certificate file is used for). In each of these, the key is used for Certificate Signing. According to IBM's *Key usage extensions and extended key Usage* page, the Certificate Signing key usage extension is to be used when the subject public key is used to verify a signature on certificates. This extension can be used only in CA certificates.[53] We summarize the contents of the Wuala CA certificate below:

```
Alias name: wualaca, Creation date: Jan 6, 2012
Entry type: trustedCertEntry, Owner: CN=Wuala CA, OU=Wuala,
EMAILADDRESS=cert@wuala.com, Issuer: CN=Wuala CA, OU=Wuala, O=LaCie
EMAILADDRESS=cert@wuala.com, Signature algorithm name: SHA1withRSA
```

As indicated by the algorithms identified in the certificates as well as the key usage

extension, this certificate is used for creating and verifying digital signatures. However, the same certificates could be issued by Wuala for performing other PKI-related functions to include both encryption and decryption of user data or the creation of private groups.

3.4.2 Spider Oak Analysis

In the case of Spider Oak, the disassembly of its binary did not yield results to substantiate our claims, thus we focused primarily on the results of decompilation and network traffic analysis in this section. During the code analysis of Spider Oak, we uncovered a 'Public Key' folder containing DSA files, RSA files, a public key file, and other crypto-related files. These files were each disassembled and evaluated for evidentiary purposes. The 'RSA.pyc' file is presented below as a representative sample. It shows a call to (or definition of) a 'generate' function used to generate an RSA public/private key pair programmatically. There is a similar file for generating a DSA key pair which we excluded. The generation of certificates in a programmatic function confirms that users are not allowed to use their own certificates for cryptographic purposes which can allow Spider Oak to generate certificates 'on behalf' of legitimate users. As a result, a user could be tricked into sharing data with Spider Oak instead of the intended user.

```
db "generate(bits:int, randfunc :callable, progress_func:callable)
Generate an RSA key of length 'bits', using 'randfunc' to get
random data and 'progress_func', if present, to display the
progress of the key generation. l\x02"
```

Unlike Wuala, Spider Oak does not appear to use any form of code obfuscation. The files we examined contained the .pyc extension which is the equivalent of a Java Byte Code file for the Python programming language. To analyze these files, we had to decompile them to the original Python Source code. Similar to Wuala, Spider Oak uses a file called 'certs.pyc' to store its certificates. This file contained 5 certificate entries: Equifax Secure CA, GeoTrust Global CA (2), RapidSSL CA, and the Spider Oak Root CA. Contents of the Spider Oak Root CA certificate are as follows:

```
Data:  Version: 1 (0x0), Serial Number: ea:14:d7:ad:6a:cf:dc:35
Signature Algorithm: sha1WithRSAEncryption
Issuer: emailAddress=ssl@spideroak.com, organizationName=SpiderOak
```

This analysis phase revealed a self-signed root certificate from Spider Oak. A self-signed certificate is an identity certificate that is signed by the same identity it certifies. As mentioned above, a typical PKI scheme requires certificates to be signed by a Trusted Third Party (TTP) for verification purposes. Self-signed certificates cannot (by nature) be revoked, which may allow an attacker who has already gained access to monitor and inject data into a connection to spoof an identity if a private key has been compromised.[54] In this case, the attacker could potentially be Spider Oak (e.g., spoofing the identity of another user and gaining access to user data).

Lastly, examining the network traffic between the Spider Oak client and server produced the same root certificate identified during Decompilation. As a root certifier, Spider Oak is also responsible for validating any certificate in its trusted certificate

chain – to include Spider Oak client certificates issued to users. Finding this type of certificate also demonstrates the fact that as a certificate issuer, Spider Oak can issue user certificates that spoof the identity of another user resulting in a data confidentiality breach when data is shared with another Spider Oak user. Similar to the Wuala example (discussed above), the Spider Oak Client certificate is also validated by its respective Spider Oak Root CA.

In a recent article, Eric Snowden urges consumers to adopt more secure file storage systems which are less susceptible to government surveillance - mentioning Spider Oak specicially.[55] However, he fails to disclose (or realize) the fact that data confidentiality could be breached if data is shared (as shown above) within these same environments. Additionally, Spider Oak was interviewed by Senior Writer Brian Butler of Network World concerning some of our findings.[56] The company claims that the features we evaluated fall outside of the scope of features that they have implemented,[57] however, we assert that in order to share encrypted data - some form of Public Key Encryption must be used. As a result, Spider Oak users are still subject to this type of flaw.

3.4.3 Tresorit Analysis

During the Decompilation of Tresorit, we did not identify any independent certificate files as was the case with Wuala and Spider Oak. According to Tresorit's documentation, a roaming profile file contains both the SSL/TLS client certificate and the associated private key required to communicate with Tresorit's servers, and

agreement certificates and private keys to securely access and decrypt the contents of a Tresor. Hence, there were not any specific certificate entries to examine during this phase.

Disassembly of the Tresorit binary contained a number of references to certificate operations, files, and uses with application. The entry below shows the loading of a certificate in the traditional X.509 format. This confirms that X.509 certifications are utilized in the Tresorit client application for cryptographic functions, however, does not fully substantiate the fact that Tresorit is operating in a Certificate Authority (CA) capacity.

```
db "y_OBJ", 0, db  "X509_get_pubkey_parameters", 0
db "X509_load_cert_crl_file", 0, db "X509_load_cert_file", 0
db "X509_load_crl_file", 0
```

In the network analysis of Tresorit, we identified a certificate issued by Tresorit — the Tresorit User CA certificate. This certificate differs from the Wuala and Spider Oak certificates in that it is validated by the StartCom CA which would imply that they are using a TTP to ensure 100% data confidentiality during sharing transactions. However, the fact that this certificate was issued by Tresorit verifies that they also can view user data whenever the keys are used for encryption/decryption purposes. Additionally, the fact that users are neither allowed to use their own certificates or enter credentials to generate their own certificates upon signing up with any of the three CSPs gives credence to our claims that counterfeit certificates could be issued. This would lead users to implicitly trust that other users are who they say they are

(i.e., the CSP is not acting on their behalf). In the case of Tresorit, StartCom CA is serving as the Root CA with Tresorit serving as an Intermediate CA. This differs from both Wuala and Tresorit, however, is still indicative of Tresorit operating as a CA in some capacity.

In this chapter we examined three major Cloud Storage Providers (CSPs) for weaknesses associated with their sharing functions. Each CSP claims to offer 100% user data confidentiality through all data transactions executed within their respective cloud instances. We discovered that these principles of confidentiality hold true in use cases where data is not shared with other cloud users or with entities outside of the cloud storage environment (e.g., non-members). We presented several scenarios in which data confidentiality could be breached to include: when data is shared via Public Web Link, when data is shared via Private Web Link, when CSP is accessed via Web Browser (i.e., web access), and when data is shared via Group or Private folder (due to the CSP being the issuer of the certificates used for cryptographic operations – hence could allocate counterfeit certificates to users). We provide evidence in the form of Network Traffic and Source Code analysis for each CSP to substantiate these claims. In the next chapter we provide a stronger version of the Pretty Good Privacy (PGP) certificate that can be used by users in a cloud environment to address some of the secure sharing challenges we highlight in this chapter.

Chapter 4

Enhancing PGP using Bitcoin and the Blockchain

In a recent article, Yahoo announced its intentions to add an extension that will provide its customers with the ability to digitally sign and encrypt messages using Pretty Good Privacy (PGP). Yahoo plans to use a fork of Google's End to End OpenPGP plugin that is currently in development. Yahoo follows the likes of Google, Facebook and Microsoft, who also recently announced they would encrypt internal traffic in response to the Snowden spying revelations.[58] Traditional methods of securely sharing between two or more parties rely on the use of Public-Key Encryption within a Public Key Infrastructure (PKI). In a traditional PKI scheme, a certificate authority or certification authority (CA) is an entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject

of the certificate. This allows others (relying parties) to rely upon signatures or assertions made by the private key that corresponds to the public key that is certified. In this model of trust relationships, a CA is a Trusted Third Party (TTP) that is trusted by both the subject (owner) of the certificate and the party relying upon the certificate. CAs are characteristic of many PKI schemes.[59] Currently, the most viable alternative for Public Key Crytography based on a CA is PGP. PGP is built upon a Distributed Web of Trust in which a user's trustworthiness is established by others who can vouch for that user's identity. Preventing its wholesale adoption are a number of inherent weaknesses to include (but not limited to) the following:

- Trust Relationships are built on a subjective honor system
- Only first degree relationships can be fully trusted
- Levels of trust are difficult to quantify with actual values
- Issues specific to the Web of Trust:

1. **Certification.** It is currently difficult to get certified if the key is new. In general people complain that it is hard to find endorsers to enhance the trustworthiness of a new key - which will limit its use.
2. **Endorsement.** Competence and willingness of endorsers. There is currently no incentive to endorse a key of someone you know - much less someone you indirectly know through a friend or relative.

Bitcoin is a form of digital currency, created and held electronically.[60] According to "Crypto Coin News", the number of active Bitcoin users worldwide will reach 4.7 million by the end of 2019, marking a significant gain over the 1.3 million last year, according to a report from Juniper Research.[61] As a result of its popularity, we introduce a new Bitcoin-Based PGP certificate format, certificate validation methodology, and certificate endorsement model that overcomes the issues we have highlighted above. Issues 1 and 2 with the Web of Trust can be easily solved using our new Bitcoin-Based PGP certificate format and verification system. Issue 4 can be resolved by use of an endorsement fee. The amount of the fee can be determined by the user and will vary based on the current value of a Bitcoin - which has been relatively stable of late.[62] In Issue 2, the bitcoin payment ensures that the endorser follows the "authentication" procedure otherwise they risk losing bitcoins - which demonstrates both their competence and willingness to serve as a viable certificate endorser.

There are some interesting properties of our trust establishment protocol that could result in safer use of PGP. Property 1) People have the option of using previous transactions before using a certificate OR directly establishing a trust relationship themselves with a certificate owner (i.e., direct trust). Property 2) As mentioned above, because of the risk of losing bitcoins via the identity-verification process, people will be less likely to leverage our certificates without a direct trust establishment. Property 3) The block chain and associated identity-verification transactions provide

transparency into the trustworthiness of others. In addition to these benefits, we also provide the design of a novel PGP Key Server based on the blockchain's ability to store pieces of data since the 0.9.0 release. The 0.9.0 release of Bitcoin Core added a new standard transaction type granting access to a previously disallowed script function, *OP-RETURN*.[63] This function accepts a user-defined sequence of up to 40 bytes (now 80 bytes in current release). Once realized, this new key server will be completely de-centralized and serve as an appropriate repository for Bitcoin-Based PGP Certificates. Our work in this chapter specifically provides the following contributions:

- **Bitcoin-Based PGP Certificate:** Contains Bitcoin address for identity verification and certificate revocation.
- **Identity-Verification and Revocation Transactions:** Serves as alternative means of verifying a certificate owner's Public Key contained in a Bitcoin-Based PGP Certificate. Also provides a mechanism for certificate revocation using the embedded Bitcoin address for revocation purposes.
- **PGP Trust Levels:** Allows users to specify the amount of Bitcoins they are willing to "risk" in order to verify a particular Bitcoin-Based PGP certificate. This amount (determined by the verifier) now attests to a level of trust the verifier has in the certificate owner.
- **Certificate Signing Endorsements:** Adds small incentive fee (via Bitcoin)

each time an endorser (with a valid Bitcoin address) signs an Enhanced PGP Certificate stored on Bitcoin-Based PGP Key Server.

- **Bitcoin-Based PGP Key Server Design:** Demonstrates method of using the Bitcoin Transaction Blockchain for PGP Key Storage. It offers a completely decentralized client-based software application that allows users to efficiently store and retrieve Bitcoin-Based PGP certificates within the blockchain. The application will separate certificate into individual pieces to fit within the allowed number of bytes and store within blockchain - as append only data. Similarly, upon request (e.g., based on PGP Key ID or similar), the client will facilitate the retrieval of PGP certificate fragments and reassemble them for use by the requesting user.

4.1 Related Work

This section examines previous work that proposes novel methods to protect the confidentiality of data in an uncontrolled network-accessible environment without the aid of a Certificate Authority (CA). According to,[64] BitPay has launched a project that leverages bitcoin technology to facilitate a decentralized authentication system. Called BitAuth, the system uses cryptographic signatures in place of server-side password storage. BitAuth is a way to do secure, passwordless authentication using the same elliptic-curve cryptography as Bitcoin. Instead of using a shared secret, the

client signs each request using a private key and the server checks to make sure the signature is valid and matches the public key. A nonce is used to prevent replay attacks and provide sequence enforcement.[65] Similar to our novel Bitcoin-Based PGP certificate, BitAuth provides "portable" identity in that the same identity can be used with (or on) multiple services. BitAuth is a promising new method of authentication, but currenly relies heavily on the System Identification Number (SIN). The SIN is a new concept that is similar to a Bitcoin address, however, is not widely adopted. Whereas, our scheme relies on popular Bitcoin primitives - address, transactions, and the block chain - that are widely being used. Additionally, since the focus of BitAuth is on authentication, it cannot be used to protect the confidentiality of information shared between two parties - as is the primary benefit of our Bitcoin-Based PGP Certificate.

Off-the-Record (OTR) Messaging is a protocol designed for private social communications. According to,[66,67] the notion of an off-the-record conversation captures the semantics one intuitively wants from private communication - only the two parties involved are privy to the contents of the conversation; after the conversation is over, no one (not even the parties involved) can produce a transcript; and although the participants are assured of each other's identities, neither they nor anyone else can prove this information to a third party. Current versions of the OTR protocol, support mutual authentication of users using a shared secret through the socialist minimalist protocol. This feature makes it possible for users to verify the identity of

the remote party and avoid a man-in-the-middle attack without the inconvenience of manually comparing public key fingerprints through an outside channel. OTR's primary weakness lies in the fact that it is primarily applicable in the domain of instant messaging - whereas our Bitcoin-Based PGP certificate can be used in virtually any domain in which secure information sharing is desired. According to the authors of the OTR protocol, "The high latency of email communication makes using our"off-the-record" protocol impractical in the setting of email."

In,[68] a secure replacement for CAs is proposed. Rather than employing a traditionally hard-coded list of immutable CAs, Convergence allows one to configure a dynamic set of Notaries which use network perspective to validate user communications. It provides the following guarantees: Trust Agility, Robustness, Backwards Compatibility, Extensibility and Anonymity. This all occurs within a distributed environment in which anyone can serve as a trust notary. Convergence originated from the ideas originally developed by the Perspectives Project at Carnegie Mellon University.[69] Convergence has great promise in the domain of web browser security and other areas where SSL is prominent. However, it suffers from the fact that the number of notaries currently in existence for performing CA functions is limited (due to it being a fairly new effort). As a result, this could lead to potential Denial of Service (DoS) attacks in the event the notaries become overwhelmed with requests. The Bitcoin infrastructure - upon which our certificate primarily relies - has successfully processed nearly 40 million transactions (to date).[70] This makes it robust against

volume-based security attacks such as DoS and DDoS - when applicable.

4.2 Public-Key Digital Certificates

In this section we discuss the differences between the traditional PGP Certificate and our novel Bitcoin-Based PGP certificate which leverages aspects of the Bitcoin infrastructure to address the shortcomings of the existing PGP certificate web of trust model.

4.2.1 PGP Certificates

PGP is a public key cryptographic package, which is intended for public usage. It provides sender authenticity, message integrity and non-repudiation of the sender. Although PGP can encrypt any data or files, it is most commonly used for e-mail which has no built-in security as originally implemented. It was originally designed and developed by Phil Zimmermann in 1991. A PGP certificate includes (but not limited to) the following information:[72]

- **PGP Version Number**: This identifies which version of PGP was used to create the key associated with the certificate.
- **Certificate holder's public key**: This is the public portion of the key pair, together with the algorithm of the key: RSA, Elgamal or DSA.

- **Certificate holder's information**: This is information about the user, such as his or her name, user ID, e-mail address, ICQ number, photograph, or other identifier.

- **Digital signature of the certificate owner**: This item, also called a self-signature, is the signature generated using the corresponding private key of the public key associated with the certificate.

- **Validity Period**: The certificate's start date/time and expiration date/time; indicates when the certificate will expire. If the key pair contains subkeys, then this includes the expiration of each of the encryption subkeys as well. Subkeys enable convenient use of separate keys for signing and encryption.

- **Preferred symmetric encryption algorithm for the key**: This indicates the encryption algorithm to which the certificate owner prefers to have information encrypted by.

A PGP certificate identifies the owner of the public key and contains a signature of the key's owner, which states that the key and the identification go together. Each label contains a different means of identifying the key's owner (e.g., the owner's name and corporate e-mail account, the owner's nickname and home e-mail account, a photograph of the owner - all in one certificate). A single certificate can contain multiple signatures. Several or many people may sign the key/identification pair to attest to their own assurance that the public key definitely belongs to the specified

owner. The list of signatures of each label may differ. Signatures attest to the authenticity that one of the labels belongs to the public key, not that all the labels on the key are authentic.[73] Unlike, the X.509 certificate, PGP certificates do not rely on a CA for identity-verification - but on a Web of Trust. As this 'web' grows the reputation (or binding of the certificate to the user identity therein) proportionally grows. Since PGP allows anyone to be a 'trusted introducer' on the web of trust, the identity-verification could be very subjective and easily exploited (e.g., by someone who simply signs each certificate he/she receives without verifying the requestor's identity). This is very similar to a facebook user who accepts all friend requests without confirming whether or not they know the individual who sent the request. Additionally, as mentioned above, because PGP has not been widely adopted and is difficult to deploy, we seek to leverage Bitcoin to encourage a larger scale adoption of PGP. In the following subsection, we present our Bitcoin-Based PGP certificate format that builds on this concept of a web of trust to primarily overcome some of the weaknesses inherent to the PGP.

4.2.2 Bitcoin-Based PGP Certificates

Bitcoin is an experimental, decentralized digital currency that enables instant payments to anyone, anywhere in the world. Bitcoin uses peer-to-peer technology to operate with no central authority: managing transactions and issuing money are carried out collectively by the network. Bitcoin is designed around the idea of using

cryptography to control the creation and transfer of money, rather than relying on central authorities.[74,75] Our Bitcoin-Based PGP certificate contains all the relevant elements found in a traditional PGP Certificate but also includes a Bitcoin Address for Identity-Verification and one used for Certificate Revocation. A Bitcoin address is an identifier of 27-34 alphanumeric characters, beginning with the number 1 or 3, that represents a possible destination for a Bitcoin payment. A Bitcoin transaction is a signed section of data that is broadcast to the network and collected into blocks. It typically references previous transaction(s) and dedicates a certain number of bitcoins from it to one or more new public key(s) (Bitcoin addresses).[76] Because transactions must be verified by miners, Bitcoin users are sometimes forced to wait until they have finished mining. The bitcoin protocol is set so that each block takes roughly 10 minutes to mine. In the case of a purchase, some merchants may make users wait until this block has been confirmed, which will delay the receipt of the digital goods that have been paid for - whereas in other cases (e.g., low value transactions), a merchant will give access to the goods prior to the transaction being verified by the miners.[77] In our case, the delay does not pose a major problem since it will only take place when a trust relationship is being established for the first time - not upon certificate generation. The value of using Bitcoin in the context of a PGP certificate centers around the fact that because it is built upon a peer-to-peer network, it is able to perform its functions (e.g., money transfers, double-spending prevention) without the aid of a CA - similar to the traditional web of trust. This is advantageous in any

context where end-to-end data confidentiality is needed or desired (e.g., email, text message, cloud sharing, or social network communications). Users are more likely to trust an infrastructure that is independent of any CAs, but can still offer the same cryptographic guarantees (i.e., confidentiality and integrity) as an environment that is under their full control or purview.

4.3 PGP Threats and Security Goals

In this section, we expound on the threats we identified in the introduction and describe our security goals. We make the following assumptions as it pertains to these threats.

1. The PGP users are leveraging the Web of Trust for certificate vetting.

2. The PGP user certificates are also capable of supporting a hierarchical structure and use of a CA (similar to the traditional X.509 certificate). However, we make the assumption that a CA is not being used.

3. PGP users can assign a level of trust to another PGP user's public key

4. PGP users can assign a level of validity to another PGP user's public key

4.3.1 Threat Scenarios

Although there are a number of well documented issues with PGP, we primarily focus on threats relating to certificate validation, endorsement, and trust relationship establishment. Other threats associated with PGP, such as privacy of the Web of Trust and the fair exchange of Certificate Endorsers, are outside of the scope of our research. One of the main issues with the Web of Trust model is that beyond a first degree trust relationship, it is difficult to quantify trust. As a result, it becomes a scenario that is relatively easy for an attacker to exploit. This is primarily due to the fact that within PGP, anyone can serve as a trusted introducer (essentially a CA) to another individual. This is analogous to a member of a social network accepting every friend request that he/she receives. There is no way of knowing whether or not the individual being vouched for is actually who they say they are - which makes this a very dangerous weakness of the current PGP Web of Trust model.

One can mitigate the threat of compromise through use of our new Bitcoin-Based PGP certificate and Public Key validation process. Recall that during the identity-verification process, there is always the risk of losing the Bitcoins that are sent to a Certificate Owner. However, in the event that the Bitcoins are not returned (e.g., due to an adversarial threat or greedy Certificate Owner), the more valuable commodity (the information) will be spared from being exposed. In our new certificate, since we associate a Bitcoin Value with a trust relationship, it is easy to infer (by examining previous identity-verification transactions), how much an individual is trusted. By

using Bitcoin as a form of trust establishment, as the number of transactions increase to a particular Bitcoin-Based PGP certificate, the level of trust will also increase in the associated identity.

As mentioned above, the levels of trust with the PGP are difficult to rely on with any certainty. There are 3 levels of trust that can be assigned to someone's Public Key within PGP: Complete Trust(2), Marginal Trust(1), or No Trust (0). Similarly, there are also 3 levels of validity: Valid, Marginal Validity, or Invalid. PGP requires at least 1 Completely Trusted signature or 2 Marginally Trusted signatures to establish a Public Key as Valid.[72] Even with its numerical value system that maps to a particular "level", PGP users are subject to the loose definitions of validity and trustworthiness of a PGP key. The highest level of trust in a key, implicit trust, is trust in a Certificate Owner's own key pair. PGP assumes that if you own the private key, you must trust the actions of its related public key.[72] However, beyond one's own key pair, how can an actual value be assigned to a trust relationship? This challenge arises when users are asked to make security decisions. In a recent article, Michigan State University professor Rick Walsh studies the reasoning process behind the decisions people make that lead to computer security breaches.[71] His research shows that how people visualize and conceptualize hackers and other cybercriminals affects their cybersecurity decision-making. In the case of this threat, most users will assume that the mere fact that they are using PGP makes them secure (i.e., their visualization of the adversary). Hence, their concern of assigning an appropriately

level of validity or trust to another's PGP key is unlikely. It is also likely the case that there is a lack of understanding of what an appropriate designation would be - unless training were provided or there was a physical trust relationship between verifier and Certificate Owner.

With our new endorsement process offered via Bitcoin, this threat would be mitigated by the following constructs of our scheme: 1) Certificate Signing MUST precede the incentive fee. A fixed fee of 0.001 BTC is sent to the Bitcoin address provided by the certificate endorser (fee is paid from the certificate owner's bitcoin address - as available). This fee cannot be modified by the certificate owner OR the endorser - however, it serves as a small incentive (e.g., Thank You for Signing) to willing and competent endorsers, 2) Endorsement process is not automated. Our prototype forces users to go through a step by step process in order to sign a certificate stored on our server, and 3) Levels of Trust are established by the certificate endorser, not certificate owner. In our scheme, when performing an identity-verification transaction, any amount of Bitcoins can be sent for verification purposes. These Bitcoins are 'at risk' until the certificate owner returns them. As a result, this serves as a very clear indication of trust between certificate endorser and owner. For example, one would risk more Bitcoins in verifying a close friend's PGP certificate than someone who was not in his/her close circle of friends. As a last option, subsequent endorsers of a particular PGP key can glean levels of trust very easily prior to 'risking' their own Bitcoins using an identity-verification transaction - as previously discussed.

A few additional threats to consider with leveraging Bitcoin as an alternative method of certificate verification are those related to rogue certificate owners, wealthy endorsers, and untrustworthy endorsers. In the first case, a certificate owner can generate a PGP key and use it for collecting payments and never return incoming identity-verification transactions to endorsers. To further complicate this scenario, a wealthy endorser risks very little by endorsing such users. To address these threats, we still rely on the PGP trust model that allows for out-of-band methods of certificate verification and a web of trust. The inference is that users will not initiate an identity-verification transaction with someone they do not already know and trust from prior interactions. Additionally, in the case of the wealthy endorser, only one verification transaction is considered valid for a particular certificate. Thus, their credibility (over time) will come into question as they continue to endorse untrustworthy certificates. Lastly, we consider the scenario where endorsers are suspected of being malicious by endorsing 'just for the sake of endorsing'. Since our endorsement scheme does not invalidate - but augments - the endorsement process provided by PGP, over time a malicious endorser would be classified as someone who cannot be trusted - especially if they are endorsing both questionable and legitimate certificates. A legitimate case to consider is someone who is a professional certificate endorser. Someone whose professional responsibility is to endorse new certificates has their job (and reputation) to consider if they are found to be endorsing certificates that are not legitimate - over time.

4.4 Prototype Design

In this section, we describe the implementation details of our prototype. The prototype will be a hosted web application that will resemble a traditional Public Key Certificate server. A Certificate or key server receives and serves existing cryptographic keys to users or other computer programs. The keys distributed by the key server are almost always provided as part of a cryptographically protected identity certificate containing not only the key but also 'entity' information about the owner of the key. In the case of our Bitcoin-Based PGP Certificate server, the certificates will be in the OpenPGP public key format - with the additional Bitcoin addresses for identity-verification and certificate revocation as specified in section 3. The primary motivations for creating a new certificate server are to 1) Accommodate our new Bitcoin-Based PGP certificates, 2) Enable Identity-Verification and Revocation transactions, and 3) Enable Certificate Signing Endorsements. To faciliate these "features", our certificate server provides the following functions: Generate, Revoke, Verify, and Sign. We discuss each in the following sub-sections after which we detail the design of our novel PGP Key server based on the blockchain as an area of future work.

4.4.1 Generate and Revoke

Our Bitcoin-Based PGP certificates are generated using the openpgp javacript package - which is a javascript implementation of the OpenPGP protocol. This standard is defined by **RFC4880**. This package provides functions to encrypt and sign your data and communication, features a versatile key management system as well as access modules for all kinds of public key directories. As it is designed for web-based devices, it can be used as an alternative to GnuPG, also known as GPG, which is a command line tool with features for easy integration with other applications.[79] Each Bitcoin-Based PGP certificate will contain a set of required parameters prior to generation and items that will be automatically generated by the prototype application. One thing to note is that our implementation does not require any modification of the original PGP certificate format. We leverage the key server to store both the identity-verification and revocation addresses listed in the PGP comment field.

In PGP, users can revoke their certificate if they feel like the certificate has been compromised. PGP also allows a user to designate a certificate revoker. With PGP certificates, the user usually posts the revoked certificate on a certificate server. To enable an easier revocation process for our Bitcoin-Based PGP certificate, we are able to take advantage of the Bitcoin transaction block chain for revocation purposes. The block chain is a transaction database shared by all nodes participating in a system based on the Bitcoin protocol. A full copy of a currency's block chain contains every transaction ever executed in the currency. With this information, one can find out

how much value belonged to each address at any point in Bitcoin history.[80] The immutable nature of the Bitcoin Transaction Block chain makes it attractive from a certificate revocation standpoint. To revoke a certificate, we perform a Bitcoin transaction between the two Bitcoin addresses colocated in the Bitcoin-Based PGP certificate. This transaction can only be performed by the certificate owner once authenticated.

Key revocation is arguably the most important component of any certificate-based identification system. Our implementation deliberately forces the user to make a valid Bitcoin transaction to a legitimate Bitcoin address in his possession. Alternatively, the revocation status could be stored in *OP-RETURN* fields if our decentralized approach is adopted (as explained later in the paper). Our current method, however, has an important technical advantage: It makes verification of a certificate status extremely efficient since revocation transactions will be stored in the Bitcoin Unspent Transaction Outputs (UXTO) database and propagated among all nodes automatically. The UXTO are redeemable transactions and the information on certificate status will be kept in main memory for efficient verification.

4.4.2 Verify and Sign

An identity-verification transaction is the primary mechanism by which a user can verify another user's Public Key in a Bitcoin-Based PGP certificate. We accomplish this by associating a Bitcoin Address with each generated PGP certificate. To per-

form the Bitcoin Identity-Verification transactions, we leverage the 'jsonRPCClient' library's **sendtoaddress** function. This function sends a specified amount from a server's available balance to a specified bitcoin address. It takes the following parameters:

- *bitcoinaddress* - Bitcoin address to send to.
- *amount* - Amount to send (float, rounded to the nearest 0.01).
- *minconf* - Minimum number of confirmations req'd for transferred balance.
- *comment* - Comment for transaction.
- *comment-to* - Comment for to-address.

The first step in an identity-verifcation transaction is to connect to a local or remote bitcoin instance. In the case of our web server, users are required to have an existing Blockchain account (or they can sign up for one via our site). Next, the **getBalance()** function is used to determine if there are enough funds in which to initiate an identity-verification transaction. If so, the first transaction, 'Tx-1', which represents the initial identity-verification transaction, is sent to the Certificate owner's Bitcoin Address. Once 'Tx-1' is received by the Certificate owner, the Certificate owner has the option to send the funds back to the verifier in transaction 'Tx-2'. Lastly, once 'Tx-2' is received by the verifier, the transactions 'Tx-1' and 'Tx-2' are compared for equality. If they are equal, the verifier can proceed to confirm the

validity of the certificate and use it for regular PGP certificate operations. Once the identity-verification transaction is complete, the verifier will have the option of signing the Public Key of the Certificate that was just verified. This function mirrors the traditional signing of a Public Key, but differs in the fact that the signature will be added to the list of Web of Trust signatures associated with the Public Key signed. It also results in an endorsement for the signer - which is automatically sent to the signer's bitcoin address. The entire certificate endorsement process between a user Alice and a user Bob is shown in below. As shown, this entire process is conducted via our Bitcoin-Based PGP Key Server - making it very easy to endorse and verify new PGP Keys. The server also supports the use of PGP keys for normal operations such as encryption and decryption of standard ASCII text. After each identity-verification transaction and certificate signing (endorsement) operation, we provide a link to the transaction hash for external verification purposes (i.e., via the blockchain itself).

4.4.3 Blockchain PGP Key Server

In this subsection we describe the design of a novel PGP Key Server based on the Bitcoin Transaction Blockchain. Historically, the use of bitcoins' blockchain to store data unrelated to bitcoin payments has been a controversial subject. Many developers consider such use abusive and want to discourage it. Others view it as a demonstration of the powerful capabilities of blockchain technology and want to encourage such experimentation. Those who object to the inclusion of non-payment

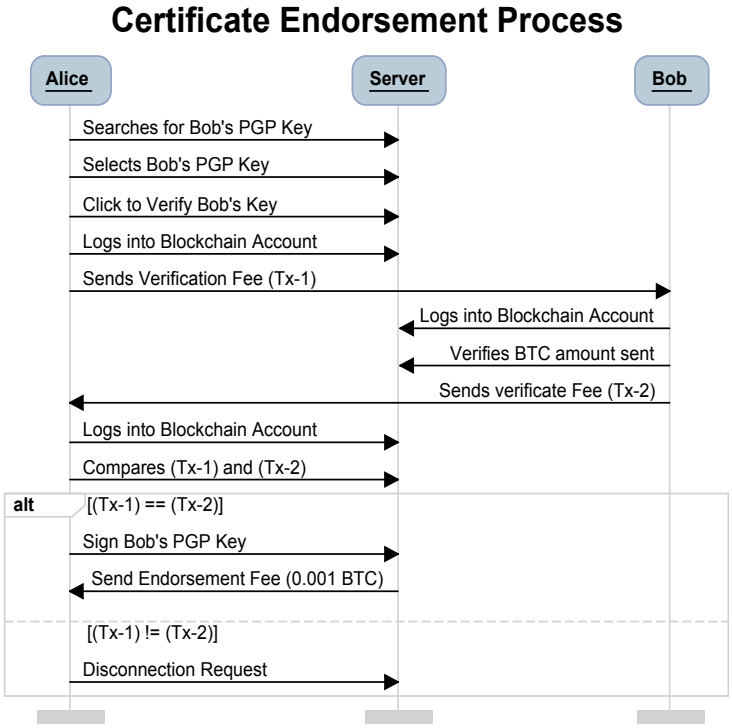

Figure 4.1: Certificate Endorsement

data argue that it causes "blockchain bloat", burdening those running full bitcoin nodes with carrying the cost of disk storage for data that the blockchain was not intended to carry. Moreover, such transactions create UTXO that cannot be spent, using the destination bitcoin address as a free-form 20-byte field. Because the address is used for data, it does not correspond to a private key and the resulting UTXO can never be spent.[81] As a result, these transactions continue to increase the size of the blockchain over time.

In version 0.9 of the Bitcoin Core client, a compromise was reached with the introduction of the *OP-RETURN* operator. *OP-RETURN* allows developers to add 40 bytes (now 80 bytes) of nonpayment data to a transaction output. However,

unlike the use of "fake" UTXO, the *OP-RETURN* operator creates an unspendable output, which does not need to be stored in the UTXO set. *OP-RETURN* outputs are recorded on the blockchain, so they consume disk space and contribute to the increase in the blockchains' size, but they are not stored in the UTXO set and therefore do not bloat the UTXO memory pool and burden full nodes with the cost of more expensive RAM..[81] As a result, innovative decentralized applications such as the one subsequently described can be realized. We focus our design discussions on storage and retrieval methods we would employ in an actualization of a prototype application and summarize with an overall component diagram.

STORAGE: Depending on the size of PGP key generated, the size could range from 1018 bytes (1024-Bit key) to 3100 bytes (4096-Bit key). PGP supports up to an 8192-Bit key which corresponds to an even larger on-disk or memory capacity for storage purposes. Keeping this in mind, along with the fact that the blockchain only accepts 'data' transactions of up to 80 bytes in size, our storage leverages an innovative certificate fragmentation mechanism to enable both logical storage and efficient retrieval. A message within our PGP Key Server will consist of a 5 Byte Header which will include the PGP Key ID (KeyID), Fragment ID (FID), Total Fragments (Total), and the Message Data. Message format is shown below:

Upon initiating a storage request, a user will provide the KeyID and PGP Key (Message Data) to the client application. The Bitcoin Address associated with the PGP Key will also be extracted and provided as input - as it will be required during

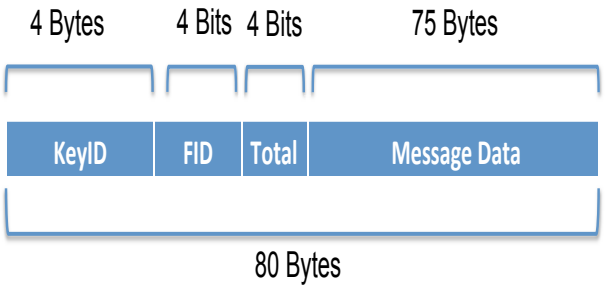

Figure 4.2: Blockchain PGP Message

the information retrieval process. The processing of each PGP Key will take place as follows:

In short, this algorithm computes Bitcoin transactions for the PGP Key 75 bytes at a time since this is the allowed size of our PGP Key Server messages (given the header of 5 bytes). A few things to note: Total Fragments denote the number of fragments there will be given the size of the PGP Certificate. In the simplest example, a PGP key of size 75 bytes will result in a Total Fragment computation of 1. The round function (or similar) will have to be used to ensure that the Total Fragment count is a whole number. The Pointer variable is used to both split and process each PGP Key segment from the beginning of the PGP Key to its end. Prior to discussing the retrieval method we employ, there are a number of similar libraries in existence that are used for similar purposes (i.e., to store data in the Bitcoin transaction blockchain. It is helpful to describe a few of the key ones here: 1) Coinspark allows you to Add messages to bitcoin transactions and Transfer any asset over the Internet, 2) Coinsecrets enables *OP-RETURN* transactions to be retrieved from the bitcoin or testnet blockchains, as well as unconfirmed transactions from the memory pool. Their

```
Input: PGP-Key, PGP-Key-Length, PGP-Key-ID, Bitcoin Address
Result: TRUE or FALSE
ChunkSize=75; Pointer=0;
Total Fragments=PGP-Key-Length / 75;
Index=0;
Result=FALSE;
Fee=0.001;
while (Pointer LESS THAN PGP-Key-Length) do
    DATA = readKey (PGP-Key, Pointer, ChunkSize);
    if more DATA then
        Header = [PGP-Key-ID + Index + Total Fragments];
        Send-Message (Header, DATA, Bitcoin Address, Fee);
        Index = Index + 1;
        Pointer = Pointer + Chunksize;
        Process Next Chunk (Continue);
    else
        Done Processing;
        Result=TRUE;
    end
    Return Result;
end
```

Algorithm 1: Process PGP Key

API also parses the content of the *OP-RETURNs* for display in decoded form, and 3) Factom is leveraged by Businesses and governments to simplify records management, record business processes, and address security and compliance issues.[82–84]

RETRIEVAL: The Retrieval of a PGP Key from the blockchain requires several steps that mirrors (in reverse) the storage operations performed. Similar to the defragmentation process of an IP datagram. The steps are as follows:

1. User requests Bitcoin-Address, KeyID from Client application (Bitcoin-Address is used to identify *OP-RETURN* transactions associated with KeyID.
2. Application Searches Blockchain for *OP-RETURN* transactions associated with Bitcoin-Address (using the *chain.com* API) and stores transactions in Results Array.
3. Application Extracts KeyID from Element[0] of Results array element
4. Application Verifies that KeyID matches KeyID from Step 1
5. Application Extracts Total Fragments from Element[0] of Results array element
6. Application Verifies number of transactions (Results Array Length) retrieved equals Total Fragments in PGP Key Message
7. If ALL transactions were retrieved successfully, Application reorders Messages and stores Messages in Ordered-Results Array
8. Reassemble PGP Key from Ordered-Results Array

9. Utilize PGP Key for Normal Operations (Sign, Verify, Encrypt, Decrypt)

Chain's enterprise-grade block chain API makes it easy to build Bitcoin applications that are fast, reliable, and secure.[85] The Get Bitcoin Address *OP-RETURN* function returns any *OP-RETURN* values sent and received by a Bitcoin Address in an Array of *OP-RETURN OBJECTS*. The *OP-RETURN OBJECT* is a pseudo-object that is extracted from the Transaction Object. It is an interpretation of the *OP-RETURN* script within a zero-value output in a Bitcoin transaction. The *OP-RETURN* can be used to include 80 bytes of metadata in a Bitcoin transaction. Each *OP-RETURN OBJECT* Contains the following data that we leverage for step 2 above. Lastly, it is important to note that the *OP-RETURN* data will have to be decoded prior to being used. This decoding process is elegantly handled by the chain API. This decoded text is the content of our message that we specify in step 2 above. We conclude this section with a overview diagram (Figure 2) of how each PGP message fragment looks within the overall structure of the Bitcoin Transaction Blockchain. It is highly probable that each message could be contained in a disparate block within the transaction blockchain. Thus the header is vital in ensuring the retrieval process works seamlessly. In Figure 2, we demonstrate a sample message broken into six message fragments and stored in three separate transaction blocks.

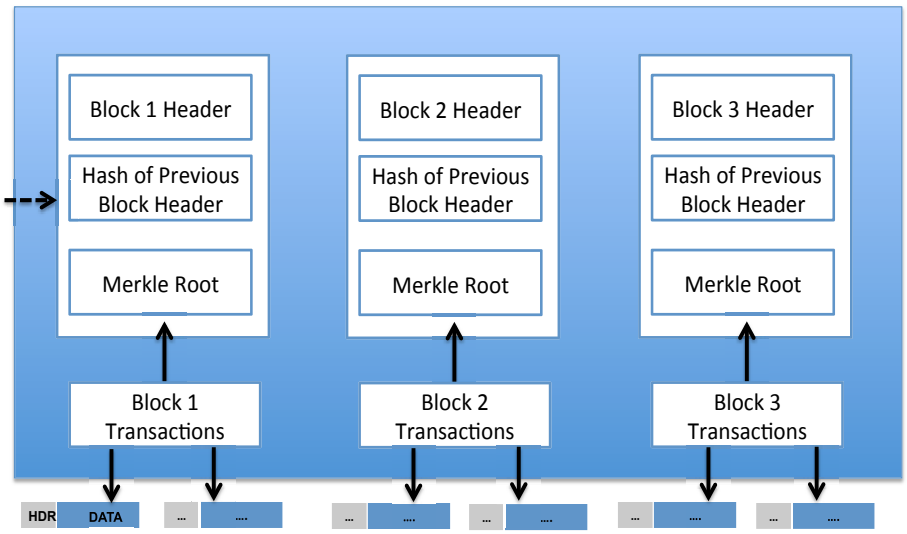

Figure 4.3: PGP Message Stored in Transaction Blockchain

4.5 Prototype Demonstration Output

In this section, we walk through the steps of using our prototype application to perform the bitcoin identity-verifications functions and the output produced by our application at each stage (as applicable).

1. Alice (Certificate Owner) Generates Bitcoin-Based PGP Certificate:

```
Key-Type: RSA
Name-Comment:
1Lk3XuR3dvPebRS6QgmVXVBjm7NBkuTuM7|
19X3kcrYNFhaPdwpdwnnBrDSNtecxUqDrW
Passphrase: pwd
Name-Real: Alice
Name-Email: alice@bitcoinpgp.com
Key-Length: 2048
%commit
```

2. Alice sends Certificate to Bob (no output - performed out of band)

3. Bob sends Identity-Verification transaction to embedded Bitcoin Address

```
Current Balance is: 0.033673
How much would you like to send for
VERIFICATION TRANSACTION? 0.00856179
Sending 0.00856179 to
1Lk3XuR3dvPebRS6QgmVXVBjm7NBkuTuM7
```

4. Alice sends Identity-Verification funds back to Bob's Bitcoin address

```
Current Balance is: 0.01198579
How much would you like to send for
RETURN VERIFICATION TRANSACTION?
0.00856179 Sending 0.00856179 to
1HiLRA7pC3d6mLGUpuhx3qH4G8sDVbdpD2
```

5. Bob checks for return of Identity-Verification funds

```
Amount Sent: 0.00856179
Transaction of 0.00856179 Found!
Result = True
```

6. Bob checks Alice's certificate for validity (result true)

7. Alice (at later date) revokes her certificate

```
Sent Revocation Transaction of Amount:
0.000017 to Revocation Address:
19X3kcrYNFhaPdwpdwnnBrDSNtecxUqDrW
1Lk3XuR3dvPebRS6QgmVXVBjm7NBkuTuM7
Certificate Successfully Revoked
```

8. Bob checks Alice's certificate for validity

```
Bitcoin Address Verified
Amount Sent: 0.000017
Revoke Transaction of 0.000017 Found!
```

In this paper we presented a number of enhancements to PGP and associated Web of Trust - which has suffered from a littany of issues since its inception. Specific issues of certification, endorsement, and ambiguous levels of trust have prevented its wide scale adoption. The contributions we discussed include a novel Bitcoin-Based PGP Certificate format, Design of a Distributed PGP Key Server - using the Bitcoin transaction blockchain for certificate storage and retrieval, Certificate Signing Endorsements, and Identity-verification and revocation transactions using Bitcoin. We demonstrate how these added features and capabilities can potentially result in a greater use of PGP which has inherently proven security properties. Our Enhanced PGP Key Server is currently hosted at [http://enhancedpgp.com/]. The next chapter seeks to address the threat of data exfiltration in Software-as-a-Service (SaaS) Cloud Storage environments to build on work we presented in this chapter.

Chapter 5

Mitigating Data Exfiltration in SaaS Clouds

5.1 Introduction

According to,[86] security incident handling, an integral part of security management, treats detection and analysis of security incidents as well as the subsequent response activities (i.e., containment, eradication, and recovery) as outlined below:

- Detection: Discover indicators of possible security incidents
- Analysis: Ascertain that indeed a security incident is at hand and understand what exactly has happened/is happening
- Containment: Contain the incident before it spreads and overwhelms resources

or increases damage

- Eradication and Recovery: Eliminate system changes caused by the incident and recover normal operations

- Preparation and Continuous Improvement: Set up/adapt incident handling activities according to changing requirements and system/threat landscape.

Existing processes and methods for incident handling are geared towards infrastructures and operational models that will be increasingly outdated by cloud computing. Research has shown that to adapt incident handling to cloud computing environments, cloud customers must establish clarity about their requirements on Cloud Service Providers (CSPs) for successful handling of incidents and contract CSPs accordingly. Secondly, CSPs must strive to support these requirements and mirror them in their Service Level Agreements. Typically, security incidents will cross the boundaries of both customer and CSP in each of the common deployment models (i.e., SaaS, Paas, and IaaS) denoting both joint responsibility and access. This must be taken into account when setting up incident handling or prevention for a cloud infrastructure. Our research focuses on attacks confined to the SaaS deployment model due to the limited level of control a user has over the security of their data once it is stored within the cloud.

Intrusion Detection Systems (IDS) have been widely used to detect malicious behaviors in network communication and hosts. IDS management is an important

capability for distributed IDS solutions, which makes it possible to integrate and handle different types of sensors or collect and synthesize alerts generated from multiple hosts located in the distributed environment. Facing new application scenarios in Cloud Computing, the IDS approaches yield several problems since the operator of the IDS should be the user, not the administrator of the Cloud infrastructure. Extensibility, efficient management, and compatibility to virtualization-based context need to be introduced into many existing IDS implementations. Additionally, the Cloud providers need to enable possibilities to deploy and configure IDS for the user. To date, a number of theoretical frameworks have been proposed to address these concerns.[87–102] None, however, have been implemented in practice, hence, the concepts that are presented cannot be effectively validated. Other research focuses on protecting the Virtual Machine (VM) within the cloud infrastructure from attacks - primarily focused on monitoring VM resources to maximize utilization or prevent Denial of Service attacks.[103–112]

One major gap identified in our assessment of the current state of affairs in applying IDS concepts in a Public cloud setting is data exfiltration detection and prevention. Data exfiltration is defined as the unauthorized transfer of data from a computer. To address this particular gap, we propose a novel method for detecting and preventing data exfiltration using cyber deception.[113] Cyber deception is a deliberate act to conceal activity on a network, create uncertainty and confusion against an adversary's efforts to establish situational awareness and to influence and

misdirect adversary perceptions and decision processes. To accomplish this, we focus on the Detection and Containment aspects of the incident handling process. In the case of our research, containment refers to the efforts aimed at preventing the exfiltration from occurring - not preventing the spread of a cyber infection. Our approach is implemented such that cloud users do not have to rely on the protection mechanisms offered by the Cloud Storage Provider (CSP) which are often limited to common Data Loss Prevention techniques - which operate based on pre-defined policies. The limitations of these approaches include: 1) can be bypassed by the use of encryption 2) are also under the control of the CSP (client-side or server-side) and 3) are not designed to prevent against unknown attacks (i.e., that are not specified in the deployed policy). To address these and other limitations we propose the **Honey Cloud Project** under which will we will address a number of these issues through Decoy Objects, Data Masking, and a Threat Aware Deception-based Cloud Infrastructure. The presented approach is not designed to replace existing solutions, but provide a mechanism that increases the Defense in Depth of data within a SaaS cloud environment.

5.2 Background

As mentioned above, existing research largely has focused on the protection on Virtual Machines within Cloud infrastructures as well as on proposing frameworks

that would aid in the detection/protection of data within the cloud. None of the aforementioned methods resulted in implementations that could be used by the public to protect data in SaaS storage infrastructures. The following research outlines a variety of methods attempting to remedy this. One common weakness with those discussed below is that the user is not in control of the protection mechanisms applied to their data. Hence, they have to trust that 1) the protections offered are adequate, 2) the audit information is accurate, and 3) the protections will not fail in the event of adversarial compromise.

5.2.1 Threat Model

When user data is stored in a Software as a Service (SaaS) Public Cloud setting, it is subject to the security features provisioned by the CSP. As mentioned above, this leads to a number of threats that user's data can be subject to without their knowledge or control. We align our efforts on the methods by which sensitive data can be exfiltrated once it has been transferred to the cloud. Our research does not examine uploads for policy conformance. Specifically we focus on data that is resident on the cloud - as we assume that users have followed any relevant policy that governs the storage of data in the cloud. Within the cloud, there are 3 primary ways data can be accessed and hence subsequently exfiltrated: 1) Via Sharing (individual or group), 2) Via download (web interface or client application), and 3) Via the cloud interface (within the cloud environment). Our research addresses the detecting and prevention

of data exfiltration via sharing and download requests. As previously stated, SaaS environments are not under the control of the user - hence we cannot modify the cloud infrastructure to monitor accesses within the environment itself. An additional feature we provide is the ability to be aware of varying threat conditions - providing a higher level of protection when the adversarial threat is at its peak.

Within the cloud setting, data exfiltration can be prevented via the following methods: 1) Detect an attempt and prevent it before it happens - would require the monitoring of all potential paths of theft in the cloud and intercepting malicious looking attempts, 2) Allow/Disallow information from being uploaded based on individual or organizational policies, or 3) Allow access to information in cloud only when given permission or authorization. In the SaaS deployment model, method 1 is not possible without the modification of the Cloud infrastructure. Method 2 focuses on protecting information from being exfiltrated via the Cloud - not after the information is stored. This has also been addressed by CloudFilter (a paper we discuss below). We aim to control data exfiltration via the 3rd method to prevent attempts by malicious adversaries who have compromised a user account.[115] We assume that users are generally trustworthy (i.e., not malicious insiders) and are able to make wise decisions about who needs to have access to their data - prior to it being uploaded. Our approach can be used as an alternative or augment to client-side encryption cloud storage vendors that offer a method of protecting user data from the cloud provider themselves.

5.2.2 Related Work

5.2.2.1 Cyber Deception:

The art of deception has long since been used as a technique to lure attackers away from important data and learn about attackers that successfully compromise a network. In the space of cloud data protection, two methods have been proposed: Fog Computing and a Honeypot implementation for the cloud. In,[116] the authors propose a novel deception-based concept that focuses on the protection of insider data theft attacks by monitoring data access in the cloud and detecting abnormal access patterns. They accomplish this by profiling the normal behavior of users and upon the detection of an anomaly, they produce and return decoy information (e.g. docs, honey files, honeypots) to the attacker. Being that there is no implementation in an actual cloud environment, it is difficult to ascertain the viability of their approach due to the fact that quantifying normal user behaviors is a difficult problem - in general - and much more in a cloud computing environment that is largely multi-tenant. Other approaches[117,118] attempt to apply anomaly-based detection mechanisms to the cloud but suffer from the same - profiling users in such a complex environment and not being implemented in an actual cloud environment.

Similarly,[119] aims to employ deception concepts in a private cloud using a honeypot. The authors deploy the Snort Network Intrusion Detection System (NIDS) in the Eucalyptus private cloud as a proof of concept implementation. They incorpo-

rate a honeypot into the NIDS to further entice attackers and provide a mechanism to learn more about similar future attacks. The primary weaknesses of their approach is that 1) it would require a major modification to existing cloud infrastructures to implement their methodologies, 2) it uses a signature-based approach for detection, and 3) their deployment would require awareness of Cloud infrastructure to enable full OS coverage (e.g. Windows, Linux, OS X). Neither the Fog Computing, Honeypot, or Anomaly-Based approaches as described would be feasible in practice due to these limitations. Our proposed solution extends the concepts present within Fog Computing by providing an implementation in an actual cloud storage environment, providing a novel mechanism for users to generate decoy information that is derived from their uploaded files, and focuses specifically on the data exfiltration problem. Additionally we provide a mechanism that can adjust to current threat conditions as we subsequently discuss.

Commercial Cloud Providers: There are a number of Commercial Cloud Providers that provide features to protect the security of cloud user data. The primary security features they all provide are: the protection of data in transit and data at rest and two-factor authentication. Other than Box, they all offer limited oversight of the activities associated with files/folders. Within Dropbox, the event log tracks when users create, delete, and restore dropbox folders. Once an action is taken, the event log records the name and ID of the dropbox folder, the action, the user who made the change, and the date it was performed.[120] For Google Drive, the types

of file activities that are recorded are the: 1) moving and removing, 2) renaming, 3) uploading, 4) sharing and unsharing, and 5) editing and commenting. There are no specific additional security features provided in the documentation.[121] SugarSync monitors any edits to a file, any new files created, files deleted, and changes to a folder or sub-folders.[122] Box[123] offers the most advanced native security features of all the existing CSPs (to include the ones described above). Box provides comprehensive reporting, logging, and audit trails in order to track account activity, file access, settings changes and nearly everything else that occurs in Box.

These features allow users to monitor for access violations only, but are not combined with a mechanism to prevent data loss. Box, however, does provide separate mechanisms to detect and prevent data loss via extended security policies that allows users to keep content confidential, mitigate data loss, flag risky sharing requests, and white list certain domains that should be trusted. Some native features are present, but others are offered via collaborative security partners. CipherCloud (one such partner) offers of a secure middleware service that provides users with insight into cloud data.[122] They provide a range of features to include: 1) User activity monitoring, 2) Anomaly Detection, 3) Comprehensive Audit Logs, and 4) Ongoing Data Loss Prevention Monitoring. This solution has a number of promising features, however, these all require trust of an entity like CipherCloud with their data (risk transference) and it also comes at a premium cost and is geared towards organizations - not individual users.

CSP IDS Implementations: In,[124] Greg Roth and Don Bailey present a detailed overview of Amazon Web Services (AWS) Intrusion Detection capabilities. These capabilities are a combination of several features: AWS's Identity and Access Management, Multi-Factor Authentication, Amazon S3 Bucket Logging, Security Audit Role, Write Once Storage for Data Provenance (Versioning), and Auditing Logs. Their approach is to use this compendium of features to detect unauthorized access to user data based on user roles. The major weakness of this approach is that it is a script based IDS, not an actual system that the general user can use. Due to the number of components involved, it would require a security expert to setup, configure, and interpret the resulting alerts. It is limited in scope - focusing primarily on access permission violations not specific attack types (e.g., data exfiltration, integrity breaches) and is specific to Amazon EC2, hence, not interoperable with other CSP infrastructures.

Lastly, in,[125] Ioannis and Pietzuch describe CloudFilter - a system for the practical control of sensitive data propagation to the cloud. Their scheme monitors HTTP File Transfers to the Cloud independent of the CSP being used. Upon upload of each user file or files, CloudFilter enforces data propagation policies to confirm that the upload request is authorized. Their approach is limited to just monitoring the HTTP protocol which would cause CloudFilter to miss sensitive data being propagating via another protocol. One major strength of the described approach is that no modification is necessary to the CSP infrastructure, however, it does require modification of files

upon upload and web browsers for user identification. This solution does not address data exfiltration once it is resident within the cloud, because it only monitors uploads.

5.3 Design Goals

As this is research is currently a work in progress, the goal of our prototype will be to provide a transparent and practical solution for protecting sensitive data stored in Software-as-a-Service (SaaS) Cloud Storage Providers, which represent stored data as files. An important requirement is for the system to be easily applicable across different cloud storage providers with minimal configuration. We will accomplish this by leveraging a cloud agnostic API to avoid having to adapt to the API of specific cloud storage service. Secondly, the deceptive information should be automatically generated based on the input provided to accompany the files that are uploaded. Lastly, we provide a solution that can be employed in a representative scenario based on the criticality of the cyber situation and/or INFOCON status. These objectives can all be accomplish without modification of the cloud infrastructure - which is key to success in a SaaS cloud environment. Based on our research, there exists no SaaS Cloud Infrastructure that leverages Cyber Deception as a means of data protection. Figure 1 provides an illustration of our prototype architecture.

A few foundational concepts in cyber deception have preceded our proposed design which we outline here for context. Honeyfiles are bait files intended for hackers

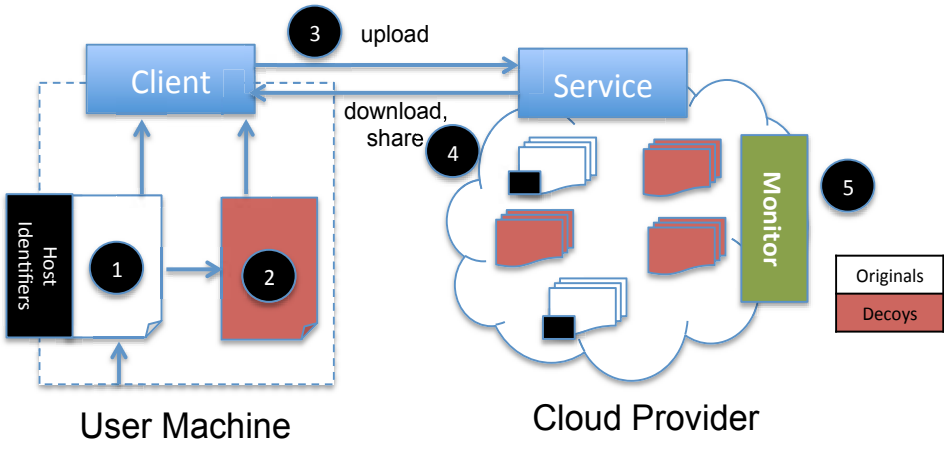

Figure 5.1: Prototype Architecture

to access. The files reside on a file server, and the server sends an alarm when a honeyfile is accessed.[127] In[128] Wang et. al. add honey activity to enhance the realism of honeyfiles. The Decoy Document Distributor (D3) System is a tool for generating and monitoring decoys. An accompanying website (FOG) allows users to download files, such as tax documents and receipts, that appear authentic but actually contain spurious information.[129] Lastly, Fox provides a mechanism for placing decoy documents in places that are more likely to get accessed.[130] As previously stated, there has been much previous work in the area of using deceptive data to defend against malicious adversaries. Yet no prior work has attempted to adapt these concepts in a SaaS cloud storage environment - which is out of the control of a user. Secondly, no work has been proposed that provides the ability to increase or decrease protections based on varying threat situations. To accomplish these objectives, we propose a novel cloud storage environment, under the Honey Cloud Project, that uses Cyber Deception to protect against unauthorized data access requests - leading to potential

data exfiltration.

Cyber threats commonly change over time within a particular environment. We argue that this is also the case within a cloud storage environment. Hence, a solution presented to protect data within that environment, should also be able to adapt to the ebb and flow of the threat landscape. As the threats increase, the associated protections to should increase as well. The converse is true for decreasing threat conditions. We describe the United States threat level system Information Operations Condition (INFOCON) levels below - which we will adapt for our purposes under the Honey Cloud Project.

- INFOCON 5 describes a situation where there is no apparent hostile activity against computer networks. Operational performance of all information systems is monitored, and password systems are used as a layer of protection.

- INFOCON 4 describes an increased risk of attack. Increased monitoring of all network activities is mandated, and all Department of Defense (DoD) end users must make sure their systems are secure. Internet usage may be restricted to government sites only, and backing up files to removable media is ideal.

- INFOCON 3 describes when a risk has been identified. Security review on important systems is a priority, and the Computer Network Defense system's alertness is increased. All unclassified dial-up connections are disconnected.

- INFOCON 2 describes when an attack has taken place but the Computer Net-

work Defense system is not at its highest alertness. Non-essential networks may be taken offline, and alternate methods of communication may be implemented.

- INFOCON 1 describes when attacks are taking place and the Computer Network Defense system is at maximum alertness. Any compromised systems are isolated from the rest of the network.

As a means of creating deceptive objects (i.e., decoy documents), we will examine a variety of data masking techniques. Data masking comes in a number of forms to include: Data Substitution, Data Shuffling, Number and/or Data Variance, Nulling or Masking Out of Data, and Data Encryption. Data Substitution allows the masking to be performed in such a manner that another authentic looking value can be substituted for the existing value. Data Shuffling is similar to the substitution method but it derives the substitution set from the same column of data that is being masked (i.e., in a database). In very simple terms, the data is randomly shuffled within the column. Number and/or Data variance varies the original value by a specified amount (e.g., +/- 10 or 20 days). Nulling or Masking Out of data either nulls a value or masks (e.g., 123-45-7455 translates to XXX-XX-XXXX). Lastly, Data Encryption is often the most complex approach to solving the data masking problem - since it is an expensive operation that requires a key. To enable legitimate cloud users' to distinguish between decoys and actual files, host system identifiers will be embedded within each file (e.g. Computer MAC Address, Network IP Address, Network Hostname).

To accommodate changes in the threat environment associated with a particu-

lar Cloud Storage Provider (CSP), we will provide a mechanism to adjust the data protection level based on threat intelligence. Table 5.1 provides and overview of the conditions imposed in a sample Honey Cloud environment. For each INFOCON level, a more sophisticated level of Data Masking can be employed and the amount of decoy information exposed can be proportionally adjusted to the current INFOCON level or criticality of a particular cyber condition. Through this work we will consider these research questions as a starting point: 1) What data masking techniques are best suited for creating decoy documents? 2) What data masking techniques should be employed at each INFOCON level? 3) Should other decoy concepts such as Endless Files or Honey Encryption be considered in the Honey Cloud? 4) Can Functional Preserving Encryption be used as a means of Data Masking? 5) What threat intelligence sources should be considered for threat level adjustments? 6) How can threat intelligence sources be incorporated into Honey Cloud environment without modifying SaaS Cloud infrastructure? 7) What is the storage requirements and overhead for the Honey Cloud environment? 8) What are the performance impacts of the Honey Cloud? 9) What information should be logged during access requests? 10) Is the Honey Cloud better suited for an IaaS or PaaS Cloud Storage Environment?

In this chapter, we have described the aspects of our Honey Cloud Project - which provides several methods to address the problem of data exfiltration detection and prevention in a SaaS Cloud Storage environment. Our proposed methods overcome the limitations of existing approaches that primarily focus on the protection of the

Table 5.1: Sample Honey Cloud Environmental Conditions

Threat Level	**Data Masking Technique**	**Deceptive Environment**
INFOCON 5	Shuffling	20% Deceptive Objects
INFOCON 4	Number & Date Variance	20% Deceptive Objects
INFOCON 3	Substitution	60% Deceptive Objects
INFOCON 2	Nulling or Masking out	80% Deceptive Objects
INFOCON 1	Encryption (FHE)	100% Deceptive Objects

virtual infrastructure - not the data that is stored in these environments. We will leverage a number of concepts in cyber deception for data protection and introduce the ability to adapt to varying threat conditions. Data Loss Prevention (DLP) is often implemented within a cloud storage environment to protect against data exfiltration. However, we found that DLP has a number of shortfalls to include: 1) can be bypassed by the use of encryption 2) are also under the control of the CSP and 3) are not designed to prevent against unknown attacks. As a result, we show the added value of using decoy documents within a SaaS environment for more dynamic data protection. The aspects of the Honey Cloud project discussed in this chapter are currently in development.

Chapter 6

Future Work

To address the weaknesses discussed in the previous sections, we propose the following as future areas of research and proof-of-concept implementations. In Secure Cloud Storage environments,A users should be issued standard certificates via a Trusted Third Party - which is typical in a traditional Public/Private Key implementation - or should be allowed to use their own certificates (e.g., PGP). Alternatively, out-of-band verification mechanisms should be used such as those provided by several implementations of off-the-record (OTR) messaging.[132] For example, the Cloud Storage Provider (CSP) Tresorit makes use of ICE which is a proprietary protocol that is designed to work in a semi-trusted environment. It does not rely solely on the trustworthiness of the certificate authority issuing the users' certificates, but also on an invitation secret and optional pairing password. Lastly, to enhance the security associated with sharing data via Private Web Link, it is possible to require

password authentication prior to listing shared files. Spider Oak employs such a strategy, although the CSP would still be required to verify that the password was entered correctly. In the future, we plan to extend our analysis of Secure CSPs by examining alternative methods of secure file sharing that do not rely on a CA for identity-verification - which will consequently overcome a number of weaknesses we have identified in this research.

In the space of leveraging user-owned certificates such as PGP in a hosted environment, our future work will consist of examining alternative methods of employing Bitcoin for identity-verification using actual Bitcoin Distributed Contracts or alternative methods that do not require modification of the original PGP certificate format. Keybase.io allows you to get a public key, safely, starting just with someone's social media username(s), but also provides other mechanisms of verifying a particular key (e.g., pgp fingerprint and bitcoin addresses).[133] In other words, it provides a compendium of online identifiers for a particular owner of a public key stored on their servers. An additional area for exploration would be to enable verifiers to leverage one or more of the online identifications provided by Keybase.io to strengthen the trust of certificate stored on our server (via their API). Furthermore, the integration of Bitcoin-Based PGP Certificates into infrastructures where secure sharing is offered (via text messaging, chat applications, and Secure Cloud Storage servers) would demonstrate their usefulness in actual environments. Lastly, a stronger form of certificate revocation should be explored that builds on the procedure we present

in our research.

Our primary goal in mitigating data exfiltration in Software-as-a-Service Cloud Storage environments is to provide a cloud agnostic proof-of-concept implementation. This implementation will conform to the design specifications outlined in Chapter 5. Additionally, we plan to integrate actual threat database sources to provide higher fidelity information prior to making a data protection adjustment (e.g. threatconnect, threatscape, enigmadatabase). We will identify other actions that could be taken upon notification of an increased threat level (e.g., increase the ratio of decoy objects in the environment or impose a data access timeout period). In this thesis, we focused on the action that was most effective in defending against data exfiltration attempts. Addressing other stages within the cyber kill chain model would be a natural evolution of our concepts. The goal would be to address different aspects of the user experience in the cloud (from a security perspective) with various cyber deception artifacts. In this work we have implicitly focused on mitigating the 'action on objectives' stage of the kill chain process. Future work would seek to address the Reconnosaince, Weaponization, Delivery, Exploitation, Installation, and Command & Control phases.

Chapter 7

Summary

In this thesis, we examined current methods of preserving the security and privacy of client data being processed and/or stored in a cloud computing environment. When it comes to cloud data protection, the methods employed can be very similar to protecting data within a traditional data center. Authentication and identity, access control, encryption, secure deletion, integrity checking, and data masking are all data protection methods that have applicability in cloud computing. In response to these findings, we have presented three primary contributions that focus on enhancing the overall security of user data residing in cloud storage environments. We have described several research outcomes that can aid in the protection of data stored within hosted environments such as the cloud. Our primary goal is to inform Cloud Storage Providers of ways to offer users a more secure storage environment. We also aim to shed light on the security practices of Cloud Storage vendors to enable users

to realize the benefits of the cloud infrastructure while protecting their data from adversarial compromise.

Bibliography

[1] Grance, T. and P. Mell. (2011). The NIST Definition of Cloud Computing. National Institute of Standards and Technology. 800(145):1-7.

[2] Winkler, V. (2011, August 8). Data security in cloud computing. Part 3: Cloud data protection methods. Retrieved from http://www.eetimes.com/design/embedded-internet-design/4218591/Data-security-in-cloud-computing-Part-3-Cloud-data-protection-methods.

[3] Roschke, S., Cheng, F., and Meinel, C. (2009). Intrusion detection in the cloud. Institutes of Electrical and Electronics Engineers. 94:729-34.

[4] Heels, E. (2011, August 4). Is Encryption the Solution to Cloud Computing Security and Privacy? Retrieved from http://www.cloudswitch.com/page/is-encryption-the- solution-to-cloud-computing-security-and-privacy.

[5] Yang, Y. (2011). Towards Mult-user Private Keyword Search for Cloud Computing. Institute of Electrical and Electronics Engineers. 76:758-59.

BIBLIOGRAPHY

[6] Ren. K., Wang, C. and Wang, Q. 2011. Towards Secure and Effective Utilization over Encrypted Cloud Data. Institute of Electrical and Electronics Engineers. 16:282-86

[7] Wang, S., Agrawal, D., and Abbadi, A.E. (2011). A Comprehensive Framework for Secure Query Processing on Relational Data in the Cloud. In W. Jonker and M. Petkovic (Eds.), Secure Data Management. Proceedings (pp. 1-18). London:Springer.

[8] Koletka, R. and Hutchinson, A. (2011). An Architecture for Secure Searchable Cloud Storage. Institute of Electrical and Electronics Engineers. 1-7.

[9] Ion, M., Russello, G., and Crispo, B. (2011). Enforcing Multi-user Access Policies to Encrypted Cloud Databases. Institute of Electrical and Electronics Engineers. 14:175-77.

[10] Li, Z., Chang, E., Huang, K., and Lai, F. (2011). A Secure Electronic Medical Record Sharing Mechanism in the Cloud Computing Platform. Institute of Electrical and Electronics Engineers. 98-103.

[11] Pagano, F., and Pagano, D. (2009). Using In-memory Encrypted Databases on the Cloud. Institute of Electrical and Electronics Engineers. 30-37.

[12] Castiglione, Aniello, et al. "A secure file sharing service for distributed computing environments." The Journal of Supercomputing 67.3 (2014): 691-710.

BIBLIOGRAPHY

[13] Sohn, Hosik, Yong Man Ro, and Kostantinos N. Plataniotis. "Content sharing based on personal information in virtually secured space." Digital Watermarking. Springer Berlin Heidelberg, 2009. 388-400.

[14] Sohn, Hosik, Yong Man Ro, and Kostantinos N. Plataniotis. "Content sharing based on personal information in virtually secured space." Digital Watermarking. Springer Berlin Heidelberg, 2009. 388-400.

[15] Kuo, Yen-Hung, et al. "Open Stack Secure Enterprise File Sync and Share Turnkey Solution." Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on. IEEE, 2014.

[16] Duarte, Eduardo, et al. "Secure and trustworthy file sharing over cloud storage using eID tokens." arXiv preprint arXiv:1501.03139 (2015).

[17] Shu, Jiwu, Zhirong Shen, and Wei Xue. "Shield: A stackable secure storage system for file sharing in public storage." Journal of Parallel and Distributed Computing 74.9 (2014): 2872-2883.

[18] Puttaswamy, K., Kruegel, C., and Zhao, B. (2011). Silverline: Toward Data Confidentiality in Storage-Intensive Cloud Applications. Symposium on Cloud Computing. 1-13.

[19] Bessani, A., Correia, M., Quaresma, B., Andre, F., and Sousa, P. (2011). Dependable and Secure Storage in a Cloud-of-Clouds. Architecture. 31-45.

BIBLIOGRAPHY

[20] Huang, R., Yu, S., Zhuang, W., and Gui, X. (2010). Design of Privacy-Preserving Cloud Storage Framework. Instituteof Electrical and Electronics Engineers. 36:128-32.

[21] E.Goh,H.Shacham,N.Modadugu,D.Boneh,Sirius:securing remote untrusted storage, in: Proceeding of Network and Distributed Systems Security (NDSS) Symposium 2003, The Internet Society, 2003.

[22] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, K. Fu, Plutus - scalable secure file sharing on untrusted storage, in: Proceedings of the 2nd USENIX Conference on File and Storage Technologies, FAST03, USENIX Association, Berkeley, CA, 2003, pp. 2942.

[23] E. Geron, A. Wool, Crust: cryptographic remote untrusted storage without public keys, Internat. J. Inf. Secur. 8 (5) (2009) 357-377.

[24] Rashid, Fahmida. “Salesforce.com Acquires SaaS Encryption Provider Navajo Systems.” Salesforce.com Acquires SaaS Encryption Provider Navajo Systems. 26 Aug. 2011. Web. 24 Dec. 2015.

[25] “CipherCloud: Enterprise Cloud Security.” CipherCloud. Web. 24 Dec. 2015.

[26] J. Li, M. Krohn, D. Mazieres, and D. Shasha. Secure untrusted data repository (SUNDR). In Proceedings of the 6th Symposium on Operating Systems Design and Implementation, pages 91-106, San Francisco, CA, December 2004.

BIBLIOGRAPHY

[27] A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten. SPORC: Group collaboration using untrusted cloud resources. In Proceedings of the 9th Symposium on Operating Systems Design and Implementation, Vancouver, Canada, October 2010.

[28] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish. Depot: Cloud storage with minimal trust. In Proceedings of the 9th Symposium on Operating Systems Design and Implementation, Vancouver, Canada, October 2010.

[29] "What Is Boxcryptor? Easy to Use Encryption for Cloud Storage." Encrypted Cloud Storage. 27 Feb. 2013. Web. 24 Dec. 2015.

[30] Viivo. Web. 24 Dec. 2015.

[31] "Cloudfogger - Free File Encryption for Dropbox and the Cloud." Cloudfogger - Free File Encryption for Dropbox and the Cloud. Web. 24 Dec. 2015.

[32] "Sookasa — Fully Integrated CASB and Cloud Security Provider." Sookasa. Web. 24 Dec. 2015.

[33] "TrueCrypt." TrueCrypt. Web. 24 Dec. 2015.

[34] A-SIT. "CCE File Encryption Using the Austrian Citizen Card." A-SIT. A-SIT, n.d. Web. 24 Dec. 2015.

BIBLIOGRAPHY

[35] Bohn, Robert. “NIST Cloud Computing Program.” Cloud Computing. National Institute of Standards and Technology, 02 Dec. 2011. Web. 06 Feb. 2014.

[36] SearchStorage. “Storage as a Service (SaaS).” What Is Storage as a Service. SearchStorage, Feb. 2009. Web. 06 Feb. 2014.

[37] Lacie. Wuala. Lacie, 01 Jan. 2014. Web. 06 Feb. 2014.

[38] Spider Oak. 100% Private Online Backup, Sync & Sharing. SpiderOak. SpiderOak, 2014. Web. 06 Feb. 2014.

[39] Tresorit. Secure File Sync and Share. Tresorit, 2014. Web. 06 Feb. 2014.

[40] Borgmann, Moritz, and Michael Waidner. On the Security of Cloud Storage Services. Stuttgart: Fraunhofer-Verl., 2012. Print.

[41] Mager, T., Biersack, E. and P. Michiardi. “A Measurement Study of the Wuala On-line Storage Service.” Peer to Peer IEEE International Conference Proceedings (2012): 237-48. Print.

[42] Kholia, Dhiru, and Przemysaw Wegrzyn. “Looking inside the (Drop) Box.” 7th USENIX Workshop on Offensive Technologies (2013).

[43] Hacker10. “List of USA Cloud Storage Services with Client Side Encryption.” Hacker 10 Security Hacker, 12 Sept. 2013. Web. 18 Feb. 2014.

[44] Hacker10. “List of Non USA Cloud Storage Services with Client Side Encryption.” Hacker 10 Security Hacker, 12 Sept. 2013. Web. 18 Feb. 2014.

BIBLIOGRAPHY

[45] Tresorit. “Tresorit: White Paper.” Tresorit, 2012. Web. 18 Feb. 2014.

[46] Wireshark Foundation. “WireShark.” Wireshark Foundation, 1998. Web. 18 Feb. 2014.

[47] Telerik. “Fiddler.” The Free Web Debugging Proxy by Telerik. Telerik, 2002. Web. 18 Feb. 2014.

[48] “AndroChef Java Decompiler.” AndroChef Java Decompiler, n.d. Web. 18 Feb. 2014.

[49] Bnony, Vincent. “Hopper.” Vincent Bnony, n.d. Web. 18 Feb. 2014.

[50] Synalysis. “Synalyze It!” Reverse Engineering and Binary File Analysis Made Easy. Synalysis, 2010. Web. 18 Feb. 2014.

[51] Froomkin, A. Michael. “1996 A.Michael Froomkin: The Essential Role OfTrusted Third Parties in Electronic Commerce.” 1996 A.Michael Froomkin: The Essential Role of Trusted Third Parties in Electronic Commerce. N.p., 14 Oct. 1994. Web. 18 Feb. 2014.

[52] Microsoft. “What Are CA Certificates?” Technet Library. Microsoft Technet, 3 Mar. 2003. Web. 18 Feb. 2014.

[53] “IBM Lotus Domino and Notes Information Center.” IBM Lotus Domino and Notes Information Center. N.p., 14 Aug. 2008. Web. 18 Feb. 2014.

BIBLIOGRAPHY

[54] "The IEEE P1363 Home Page." IEEE P1363 – Standard Specifications For Public Key Cryptography. N.p., 10 Oct. 2008. Web. 18 Feb. 2014.

[55] Kiss, Jemima. "Snowden: Dropbox Is Hostile to Privacy, unlike 'zero Knowledge' Spideroak." Theguardian.com. Guardian News and Media, 17 July 2014. Web. 13 Aug. 2014.

[56] Butler, Brian. "Even the Most Secure Cloud Storage May Not Be so Secure, Study Finds ." Network World. Network World Inc., 21 Apr. 2014. Web. 13 Aug. 2014.

[57] Fairless, Alan. "Comments on Study Citing Design Flaw That Puts Your Privacy at Risk - SpiderOak Blog." SpiderOak Blog. Spider Oak, 22 Apr. 2014. Web. 13 Aug. 2014.

[58] Saarinen, Juha. "Yahoo to Provide PGP Encryption for Mail." ITnews for Australian Business. ITnews, 08 Aug. 2014. Web. 26 Aug. 2014.

[59] Froomkin, A. Michael. "1996 A.Michael Froomkin: The Essential Role OfTrusted Third Parties in Electronic Commerce." 1996 A.Michael Froomkin: The Essential Role of Trusted Third Parties in Electronic Commerce. N.p., 14 Oct. 1994. Web. 18 Feb. 2014.

[60] Coindesk. "What Is Bitcoin?" CoinDesk RSS. Coindesk, 20 Mar. 2015. Web. 13 Aug. 2015.

BIBLIOGRAPHY

[61] Maras, Elliot. “Bitcoin Users To Approach 5 Million Mark By 2019, Juniper Research Reports - CCN: Financial Bitcoin/Cryptocurrency News.” CCN Financial Bitcoin Cryptocurrency News. CCN.LA, 17 Mar. 2015. Web. 13 Aug. 2015.

[62] Torpey, Kyle. “The Bitcoin Price Has Been Remarkably Stable Lately.” The Bitcoin Price Has Been Remarkably Stable Lately. Inside Bitcoins, 27 Feb. 2015. Web. 13 Aug. 2015.

[63] Apodaca, Rich. “OP-RETURN and the Future of Bitcoin.” Bitzuma. Bitzuma, 29 July 2014. Web. 29 Apr. 2015.

[64] Cawrey, Daniel. “BitPay Seeks to Decentralize Digital Identification with BitAuth.” CoinDesk. CoinDesk, 01 July 2014. Web. 06 July 2014.

[65] Bitpay. “BitAuth, for Decentralized Authentication.” BitAuth, for Decentralized Authentication. Bitpay, 01 July 2014. Web. 06 July 2014.

[66] Goldberg, Ian. “Off-the-Record Messaging.” Off-the-Record Messaging. OTR Development Team, 2012. Web. 25 Feb. 2014.

[67] Goldberg, Ian. Borisov, Nikita, and Brewer, Eric “Off-the-Record Communication or, Why Not to use PGP.” Zero-Knowlege Systems and U.C. Berkely, (2012): Print.

[68] Thoughtcrime Labs.“Convergence Details.” Convergence. Thoughtcrime Labs, 2011. Web. 02 May 2014.

[69] Wendlandt, Dan; Anderson, David G and Perrig, Adrian. “Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing”. Carnegie Mellon University (2011): Print.

[70] Bitcoin. “Bitcoin Charts Various Bitcoin Charts and Currency Statistics.” Bitcoin Charts. The Bitcoin Foundation, 2009. Web. 02 May 2014.

[71] Poor Decision-Making Can Lead to Cybersecurity Breaches — News — Communications of the ACM. (n.d.). Retrieved from http://cacm.acm.org/news/183571-poor-decision-making-can-lead-to-cybersecurity-breaches/fulltext. Web. 04 May 2015.

[72] Network Associates, Inc. “How PGP Works.” How PGP Works. Network Associates, Inc, 1999. Web. 04 July 2014.

[73] N. Prohic, “Public Key Infrastructures PGP vs. X.509 ”, INFOTECH Seminar Advanced Communication Services (ACS), 2005. Institute of Communication Networks and Computer Engineering University of Stuttgart

[74] Bitcoin.org. “Main Page (Overview).” Bitcoin. Bitcoin.org, 19 Apr. 2014. Web. 05 May 2014.

[75] Bitcoin.org. “How Does Bitcoin Work?” Bitcoin. Bitcoin.org, 2014. Web. 06 May 2014.

[76] Bitcoin.org. “Transacations.” Bitcoin. Bitcoin.org, 2014. Web. 06 May 2014.

[77] CoinDesk. “How Do Bitcoin Transactions Work?” CoinDesk RSS. CoinDesk, 06 Mar. 2014. Web. 02 July 2014.

[78] Sajip, Vinay. “Python-gnupg - A Python Wrapper for GnuPG.” Python-gnupg. Python Hosted, 2008. Web. 14 July 2014.

[79] The GnuPG Project. “The GNU Privacy Guard.” GnuPG. N.p., 1998. Web. 16 May 2014.

[80] Bitcoin. “Block Chain.” Bitcoin Wiki. Bitcoin, 21 Apr. 2014. Web. 15 July 2014.

[81] O’Reilly. “Chapter5.Transactions.” Mastering Bitcoin. O’Reilly, 2013. Web. 01 May 2015.

[82] Coin Sciences Ltd. (2015). Coin Secrets - *OP-RETURN* metadata in the Bitcoin Blockchain. Retrieved from http://coinsecrets.org/. Web. 03 May 2015.

[83] Chain Inc. “Chain — The Block Chain API for Developers.” Bitcoin API. Chain Inc., 2015. Web. 03 May 2015.

[84] Factom Inc. (n.d.). Factom - A Scalable Data Layer for the Blockchain. Retrieved from http://factom.org/. Web. 03 May 2015.

[85] Chain Inc. “Chain — The Block Chain API for Developers.” Bitcoin API. Chain Inc., 2015. Web. 03 May 2015.

[86] Grobauer, Bernd, and Thomas Schreck. “Towards incident handling in the cloud:

challenges and approaches." Proceedings of the 2010 ACM workshop on Cloud computing security workshop. ACM, 2010.

[87] Yassin, Warusia, et al. "A cloud-based intrusion detection service framework. Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on. IEEE, 2012.

[88] Modi, Chirag, et al. "A novel framework for intrusion detection in cloud. Proceedings of the Fifth International Conference on Security of Information and Networks. ACM, 2012.

[89] Kholidy, Hisham, and Fabrizio Baiardi. "CIDS: a framework for intrusion detection in cloud systems. Information Technology: New Generations (ITNG), 2012 Ninth International Conference on. IEEE, 2012.

[90] He, Sijin, et al. "Cloud resource monitoring for intrusion detection. Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on. Vol. 2. IEEE, 2013.

[91] Tse, Daniel WK, et al. "DATA DRIVEN DETECTION STRATEGY ENGINE FOR BETTER INTRUSION DETECTION ON CLOUD COMPUTING. (2014).

[92] Modi, Chirag, and Dhiran Patel. "A Novel hybrid-Network Intrusion Detection System in Cloud Computing. (2013).

[93] Alharkan, Turki, and Patrick Martin. "Idsaas: Intrusion detection system as a

service in public clouds. Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE Computer Society, 2012.

[94] Ficco, Massimo, Luca Tasquier, and Rocco Aversa. "Intrusion detection in cloud computing. P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on. IEEE, 2013.

[95] Ficco, Massimo, Salvatore Venticinque, and Beniamino Di Martino. "Mosaic-based intrusion detection framework for cloud computing. On the Move to Meaningful Internet Systems: OTM 2012. Springer Berlin Heidelberg, 2012. 628-644.

[96] Lo, Chi-Chun, Chun-Chieh Huang, and Joy Ku. "A cooperative intrusion detection system framework for cloud computing networks. Parallel processing workshops (ICPPW), 2010 39th international conference on. IEEE, 2010.

[97] Zargar, Saman Taghavi, Hassan Takabi, and James BD Joshi. "DCDIDP: A distributed, collaborative, and data-driven intrusion detection and prevention framework for cloud computing environments. Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on. IEEE, 2011.

[98] Ko, Ryan KL, et al. "TrustCloud: A framework for accountability and trust in cloud computing. Services (SERVICES), 2011 IEEE World Congress on. IEEE, 2011.

[99] Gul, Irfan, and M. Hussain. "Distributed cloud intrusion detection model. International Journal of Advanced Science and Technology 34 (2011): 71-82.

[100] Dhage, Sudhir N., and B. B. Meshram. "Intrusion detection system in cloud computing environment. International Journal of Cloud Computing 1.2-3 (2012): 261-282.

[101] Alharkan, Turki, and Patrick Martin. "Idsaas: Intrusion detection system as a service in public clouds." Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE Computer Society, 2012.

[102] Wang, Huaibin, Haiyun Zhou, and Chundong Wang. "Virtual machine-based intrusion detection system framework in cloud computing environment." Journal of Computers 7.10 (2012): 2397-2403.

[103] Bharadwaja, Saketh, et al. "Collabra: a xen hypervisor based collaborative intrusion detection system. Information technology: New generations (ITNG), 2011 eighth international conference on. IEEE, 2011.

[104] Wei, Jinpeng, et al. "Managing security of virtual machine images in a cloud environment. Proceedings of the 2009 ACM workshop on Cloud computing security. ACM, 2009.

[105] Nikolai, Jason, and Yong Wang. "Hypervisor-based cloud intrusion detection

system." Computing, Networking and Communications (ICNC), 2014 International Conference on. IEEE, 2014.

[106] Baraka, Hifaa Bait, and Huaglory Tianfield. "Intrusion Detection System for Cloud Environment." Proceedings of the 7th International Conference on Security of Information and Networks. ACM, 2014.

[107] Gupta, Sanchika, Padam Kumar, and Ajith Abraham. "A profile based network intrusion detection and prevention system for securing cloud environment." International Journal of Distributed Sensor Networks 2013 (2013).

[108] Gupta, Swastik, et al. "A fingerprinting system calls approach for intrusion detection in a cloud environment." Computational aspects of social networks (CASoN), 2012 fourth international conference on. IEEE, 2012.

[109] Arajo, Josenilson Dias, et al. "EICIDS-Elastic and Internal Cloud-based Intrusion Detection System." International Journal of Communication Networks and Information Security (IJCNIS) 7.1 (2015).

[110] Li, Bo, Jianxin Li, and Lu Liu. "CloudMon: a resourceefficient IaaS cloud monitoring system based on networked intrusion detection system virtual appliances." Concurrency and Computation: Practice and Experience (2013).

[111] Zhang, Yinqian, et al. "Homealone: Co-residency detection in the cloud via side-

channel analysis. Security and Privacy (SP), 2011 IEEE Symposium on. IEEE, 2011.

[112] Arshad, Junaid, Paul Townend, and Jie Xu. "An automatic intrusion diagnosis approach for clouds." International Journal of Automation and Computing 8.3 (2011): 286-296.

[113] Schneier, Bruce. "Schneier on Security." Blog. N.p., 23 Dec. 2015. Web. 24 Dec. 2015.

[114] Blogger, R. (2012, January 17). Top 10 Common Uses for the Cloud for 2012. Retrieved July 30, 2015.

[115] Wilson, Duane C., and Giuseppe Ateniese. "To Share or not to Share in Client-Side Encrypted Clouds." Information Security. Springer International Publishing, 2014. 401-412.

[116] Stolfo, Salvatore J., Malek Ben Salem, and Angelos D. Keromytis. "Fog computing: Mitigating insider data theft attacks in the cloud. Security and Privacy Workshops (SPW), 2012 IEEE Symposium on. IEEE, 2012.

[117] Nascimento, Gustavo, and Miguel Correia. "Anomaly-based intrusion detection in software as a service. Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on. IEEE, 2011.

BIBLIOGRAPHY

[118] Vieira, Kleber, et al. “Intrusion detection for grid and cloud computing. It Professional 4 (2009): 38-43.

[119] Borisaniya, Bhavesh, et al. “Incorporating honeypot for intrusion detection in cloud infrastructure. Trust Management VI. Springer Berlin Heidelberg, 2012. 84-96.

[120] “Dropbox.” Dropbox. N.p., n.d. Web. 24 Dec. 2015.

[121] “Google Drive - Cloud Storage and File Backup for Photos, Docs and More.” Google Drive. N.p., n.d. Web. 24 Dec. 2015.

[122] “Back up and Access Your Files. Using Your Existing Folder Structure.” Cloud File Sharing, File Sync and Online Backup From Any Device. N.p., n.d. Web. 24 Dec. 2015.

[123] “Enterprise-Grade Security, Visibility and Control — Box.” Box. N.p., n.d. Web. 24 Dec. 2015.

[124] “Cloud Monitoring — Database Activity Monitoring — CipherCloud.” CipherCloud. N.p., n.d. Web. 24 Dec. 2015.

[125] Roth, Greg, and Don Bailey. ”Intrusion Detection in the Cloud.” AWS Re:Invent. November 13, 2013. Accessed April 1, 2015.

[126] Papagiannis, Ioannis, and Peter Pietzuch. “Cloudfilter: practical control of

sensitive data propagation to the cloud. Proceedings of the 2012 ACM Workshop on Cloud computing security workshop. ACM, 2012.

[127] Yuill, Jim, et al. "Honeyfiles: deceptive files for intrusion detection." Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC. IEEE, 2004.

[128] Wang, Wei, et al. "Catching the wily hacker: A multilayer deception system." Sarnoff Symposium (SARNOFF), 2012 35th IEEE. IEEE, 2012.

[129] Bowen, Brian M., et al. Baiting inside attackers using decoy documents. Springer Berlin Heidelberg, 2009.

[130] Voris, Jonathan, et al. "Fox in the trap: thwarting masqueraders via automated decoy document deployment." Proceedings of the Eighth European Workshop on System Security. ACM, 2015.

[131] Green, Matthew. "Format Preserving Encryption, Or, How to Encrypt a Credit Card Number with AES." A Few Thoughts on Cryptographic Engineering. N.p., 10 Nov. 2011. Web. 24 Dec. 2015.

[132] Goldberg, Ian. "Off-the-Record Messaging." Off-the-Record Messaging. OTR Development Team, 2012. Web. 25 Feb. 2014.

[133] Krohn, Max. "Keybase." Keybase. Caroline Hadilaksono, n.d. Web. 10 Feb. 2015.

Vita

Duane Wilson currently possess a B.S. in Computer Science from Claflin University (Thesis: Monitoring and Analysis of Malicious Network Traffic over University Networks), a Masters of Engineering in Computer Science from Cornell University (Thesis: Design of Analysis Framework for Utilizing Firewall and System Logs as Source for Computer Intrusion Information), a Masters of Science in Information Security from Johns Hopkins University (Thesis: A Discretionary Access Control Method for Preventing Data Exfiltration via Removable Devices). His pursuit and completion of this terminal degree speaks to his passion for making contributions to the Computer Science and Cyber Security Bodies of Knowledge throughout the course of his future career.

Duane has spent over 13 years working in the field of Information Security beginning at the U.S. Army Research Laboratory as a contributing Network Analyst and subsequent Security Tool Researcher/Developer. In his role as a Sr. Cyber Se-

curity Engineer, he focused extensively on: Network Analyst Training, performing Security/Risk Assessments for High Value Infrastructure Components, Security Tool Evaluations, and providing recommendations to enhance the Computer Network Defense capabilities within the DoD. More recently, he was involved in the development of Advanced Cyber Security processes in the areas of Digital Forensics and Malware Analysis for the Security Operations Center of the Center for Medicare and Medicaid Services (CMS). Additionally, he provided insight into a series of test plans for the Joint Information Environment, operated and managed by the Defense Information Systems Agency. This effort is motivated by the DoDs desire to consolidate disparate data centers throughout the DoD into a Single Security Architecture. Lastly, he has also had the opportunity to serve as a guest lecturer at Alabama State University to discuss the topic of Cyber Criminals and develop educational curricula for the MD State Dept of Education.

Starting in November 2015, Duane has been serving as the Director of Cyber Security for Sabre Systems Inc. In this new role, Duane will be responsible for all of the business development activities relating to Cyber across the company. The company will focus on identifying opportunities for sole source work, Small Business Innovative Research initiatives, Broad Agency Announcements and internal research projects to offer to government and commercial clientele. To date, Duane has contributed to a 30-yr strategic plan for the Department of the Navy based on Computer Immunology, Submitted proposals on Cyber Deception, Naval Aircraft Risk/Threat

VITA

Assessments, a Cryptographic Workbench solution, Bitcoin Transaction Blockchain for Privacy Identity Management, and Cyber Resiliency for Industrial Control systems and applications (via Office of Naval Research).

Lastly, Duane has published a number of articles in reputable venues throughout his matriculation period at Johns Hopkins University. *A Discretionary Access Control Method for Preventing Data Exfiltration (DE) via Removable Devices* focuses on host-level protections for thumb drives or external hard drives. In, *"To Share or not to Share" in Client-Side Encrypted Clouds*, Duane presents his analysis of secure cloud storage providers and identifies a major flaw in the design of the sharing methodologies proposed. His last publication, *From Pretty Good to Great: Enhancing PGP Using Bitcoin and the Blockchain* presents an alternative method of validating PGP certificates for using in a hosted environment - such as the cloud. He is currently working on two additional publications: 1) *Mitigating Data Exfiltration in Software-as-a-Service Cloud Storage Environments* which leverages Cyber Deception concepts as an alternative or augment to traditional data loss and/or encryption methods of protection 2) *Deceptive Identities for Cloud Sharing* which discusses the possibility of using Cyber Deception to protect user information in the Cloud when information is shared.