Spring 2016

Tracking User Activity While Safeguarding Data From Attackers

Justin Dahmubed
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Information Security Commons](#)

Recommended Citation

Dahmubed, Justin, "Tracking User Activity While Safeguarding Data From Attackers" (2016). *Master's Projects*. 477.

DOI: <https://doi.org/10.31979/etd.2rhu-v8xv>
https://scholarworks.sjsu.edu/etd_projects/477

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

SAN JOSE STATE UNIVERSITY

Tracking User Activity While Safeguarding Data From Attackers

by

Justin Dahmubed

A thesis submitted in partial fulfillment for the
degree of Master of Science

in the

The Faculty of the Department of Computer Science

May 2016

The Designated Project Committee Approves the Project Titled

Tracking User Activity While Safeguarding Data From Attackers

by

Justin Dahmubed

APPROVED BY THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2016

Thomas H. Austin	Department of Computer Science
Robert Chun	Department of Computer Science
Ron Mak	Department of Computer Science

ABSTRACT

Tracking User Activity While Safeguarding Data From Attackers

by

Justin Dahmubed

Companies constantly look for ways to better understand customer activity on their websites. Website owners may want to be able to analyze customer activity without having to concern themselves with a government agency forcing them to reveal their information. Multiple analytical tools have been created, most notably Google Analytics.

In my thesis, I demonstrate how analytics data can be stored so that only the site owners can view the data about their customers. With my design, even the analytics site itself cannot decrypt the data after a given window of time has elapsed. The novel aspect of my design is that the analytical data can be securely stored and be protected against a government agency forcing the analytical site to release the data of all users. Tracking data is encrypted with a symmetric key that is then encrypted with a public key. The symmetric key expires after a short period of time prior to being stored in the database. After that window has elapsed, the key is no longer available. Only the website owners have access to the private key for a lengthy period of time that allows them to go back and view the data.

Acknowledgements

I would like to thank Dr. Thomas Austin for providing me with invaluable guidance and feedback these last two semesters. My thesis would not be what it is today without his tireless effort in helping me perfect my thesis. I would also like to thank my committee members, Dr. Robert Chun and Professor Ron Mak, two fabulous professors that I have had the pleasure of taking courses from and who have volunteered their time in order to ensure my success.

Last but not least, I would like to thank my parents for stressing the importance of education and that all challenges can be overcome when you have sharpened your mind through many years of schooling...

Contents

Abstract	i
Acknowledgements	ii
List of Figures	v
1 Introduction	1
1.1 Security Breaches	2
1.2 How My System Keeps Analytical Data Safe and Secure	3
2 Security Background Information	4
2.1 Symmetric versus Assymetric Encryption	4
2.2 Public Key Infrastructure	6
2.3 What is a Cryptographic Hash Function?	7
3 Related Work	8
3.1 Personally Identifiable Information of a user being leaked to third-party servers	9
3.2 UsaProxy	10
3.3 User Interest Based on Mouse Movement	11
3.4 Unobtrusively Studying User's Behavior	13
3.5 My Work	13
4 Design and Architecture	15
5 Implementation	18
5.1 MongoDB	18
5.2 ExpressJS	19
5.3 AngularJS	19
5.4 NodeJS	22
5.5 A Further Look Into The Architecture	22
5.6 What a Hacker Would See	26
5.7 How a Website Owner Accesses Data	27
5.8 SSH	28
5.9 Bringing It All Together	29

6	Experiments	30
6.1	Does tracking data get encrypted?	30
6.2	Is the tracking data capable of being decrypted?	31
6.3	How long does it take to generate the report?	32
7	Discussion	34
8	Conclusion	36
8.1	Future Work	36

List of Figures

2.1	Encryption and Decryption [1]	5
2.2	Public-key Cryptography [2]	5
3.1	Leakage of OSN ID to a Third-Party [3]	9
3.2	Leakage of Pieces of PII to a Third-Party[3]	10
3.3	System Architecture.[4]	11
3.4	Mouse trails recorded by the HTTP proxy[4]	12
3.5	Small sample of the log output produced by the UsaProxy.[4]	12
4.1	Encrypting Analytics Data	16
4.2	Generating Report	17
5.1	MVC[5]	20
5.2	Comparison of AngularJS' Popularity to Other Frameworks'[6]	21
5.3	Comparison of AngularJS' Popularity Growth Rate to Other Frameworks'[6]	21
5.4	Comparison of AngularJS' Size to Other Frameworks' [6]	22
5.5	HTTPS in the Browser[7]	23
5.6	Login Interface	25
5.7	Activity Interface	25
5.8	Encrypted Tracking Data	26
5.9	Encrypted User Data	27
5.10	Decrypted Tracking Data	28
5.11	Decrypted User Data	28
5.12	System diagram	29
6.1	Encrypted Tracking Data	30
6.2	Decrypted Analytical Results	31
6.3	# of Visitors vs Time to generate report in ms	33

Chapter 1

Introduction

My thesis is a proof of concept of a web analytics system with a primary focus on generating, storing, and safeguarding analytical data with keys to encrypt and decrypt the data. Multiple web analytics systems have been created, most notably Google Analytics[8]. Google prides itself on offering multiple features that will enable website owners to better understand the visitors of their website. Just some of the many features they offer are the number of visitors that have visited the owner's website each month, the routes they take to get to their website (e.g. through a search engine search, through clicking an advertisement, or by going directly to the website), and what devices they used to view the website. As of April 6th, 2012, over fifteen million websites use Google Analytics with more than sixty-percent of the top ten thousand sites using it and the same percentage of the top hundred thousand sites[9]. Evidently, there is already a great need to track user activity, and it is exponentially growing.

Speaking from my personal experience, I have created websites in which I have embedded multiple tracking tools into my code to understand my visitors better. The tools have provided me with a wealth of useful information about my visitors and what I need to do to optimize their experience. Therefore, my main motivation for my thesis was to come up with a prototype for how to track user activity at the same time keeping this data safeguarded from attackers.

1.1 Security Breaches

Turn on the news, and one will see a security breach occurring on major corporations every other week. Just to name one that has occurred relatively recently, an online-dating service, Ashley Madison, with the slogan “Life is short. Have an affair” had its security compromised[10]. The hackers claim to have stolen personal information about the site’s user base, and threaten to release Personally Identifiable Information (PII) such as names, home addresses, search histories, and credit card numbers of the users if the site does not comply with the hackers’ demands. In 2014 alone, people had up to one billion records of personal information unlawfully accessed[11]. This is up more than fifty percent from the previous year, and it is only getting worse. In a more global perspective, with the growing security threat of North Korea and of the Islamic State of Iraq, better known as ISIS, cybersecurity is becoming a major concern[12]. In the most closely related and recent case, a terrorist attack in California prompted the Federal Bureau of Investigation (FBI) to request that Apple Inc. create a new operating system that would bypass several security features and install it on an iPhone that the terrorists used. Though the FBI maintained that they want this new operating system for this phone only, Apple has argued that it creates a backdoor. In essence, Apple would be generating a “master” key that could be used over and over again, on any number of devices, jeopardizing the privacy of Apple’s users. Therefore, the need to store user information safely and securely is at an all-time high.

One of the biggest groups for maintaining information security is the US government’s National Security Agency, NSA. In an article published back in 2013, the NSA tried to target an online anonymity network called Tor[13]. NSA performed a successful attack against Tor by exploiting its browser bundle, which identified Tor users on the Internet and then executed an attack on the Firefox web browser. By using powerful data analysis tools, the NSA sifts through Internet traffic looking for Tor connections. The article delves deeper into how they exploit the group, but more importantly, it emphasizes the need to track user activity for the safety and security of the national population.

1.2 How My System Keeps Analytical Data Safe and Secure

My system places an emphasis on keeping data safe and secure and only allowing authorized personnel to view the data. The symmetric key encrypts the data, then a public key encrypts the symmetric key. The symmetric key expires after a week prior to being stored in the database. This is to prevent a hacker or a government agency from having sufficient time to hack the data even if they get access to the system. Therefore, if time runs out before s/he can decrypt the data, the symmetric key will be stored in the database, a new symmetric key pair will be generated, and it would be the attacker's job to hunt for the new symmetric key. However, the old keys are encrypted by the site owner's public key, who therefore can access all past records with ease. Critically, the website owner's private key never leaves his/her system so the analytics site never has an opportunity to reveal it.

The benefit is that the system is incapable of divulging information about the website's visitors. Also, with the private key living with the website owner, s/he can view the data indefinitely because the private key is always at the owner's disposal. The downside is that the website owner needs to manage the private key, so that if the attacker manages to get a hold of the private key, they will just need to get a hold of the corresponding symmetric key to be able to decrypt that portion of the analytical data defeating the purpose of the analytical system.

Existing analytical systems do not place enough emphasis on what happens after the attacker has been able to get into the system, which could result in the attacker compromising the system and getting a hold of the data. My system stores the data in a manner so that even if the attacker manages to get inside the system, they would not be able to get a hold of the plaintext data they are looking for.

Chapter 2

Security Background Information

The goal of my system is to provide analytical information to the website owner using the system without jeopardizing its security. Even if an attacker tries to break into the system, the website owner would still have the data. This is all done through the process of encryption. Wikipedia gives a sound definition of what encryption is[\[14\]](#). They say it is

“the translation of data into a secret code. Encryption is the most effective way to achieve data security. To read an encrypted file, one must have access to a secret key or password that enables s/he to decrypt it. Unencrypted data is called plaintext; encrypted data is referred to as ciphertext.”

Figure [2.1](#) provides further illustration of what happens when plaintext is encrypted with a key and when the ciphertext is decrypted with a key.

2.1 Symmetric versus Assymmetric Encryption

There are two types of encryption: asymmetric encryption and symmetric encryption. Asymmetric encryption is more commonly known as public-key encryption. Symmetric encryption relies on one key and that same key is used to encrypt the data as it is used to decrypt the data. Asymmetric encryption on the other hand uses two keys, a public key known to everyone and a private key known only to

FIGURE 2.1: Encryption and Decryption [1]

FIGURE 2.2: Public-key Cryptography [2]

the recipient of the message. When Alice wants to securely send a message to Bob, she uses Bob's public key to encrypt the message. Bob then uses his private key to decrypt it. One must note that the public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt them[15].

Asymmetric encryption is depicted in Figure 2.2.

Symmetric encryption is comparably fast compared to assymetric encryption[16]. This is because it does not require the complicated math that it is required of assymetric encryption. Symmetric encryption's main disadvantage is having to share the key with the recipient, since only one key is used. One must have to be astute in order to privately share the key with the recipient without an attacker getting a hold of it and ultimately compromising the system. This is because if the attacker gets a hold of the key, s/he can not only decrypt the messages sent from the sender to the recipient, but encrypt messages as well since the same key is being used for encryption/decryption. This is not the case for assymetric public-key encryption, since getting a hold of the private-key can decrypt messages sent from person A to person B, but cannot decrypt messages being sent from person B to person A, since a different key pair is used for that.

2.2 Public Key Infrastructure

A public key infrastructure (PKI) creates digital certificates that map public keys to the rightful owner(s), securely stores the certificates in a central repository, and negates them if needed.

A PKI consists of:

- A *certificate authority* (CA) that stores, issues and signs the digital certificates
- A *certificate management system*
- A *certificate policy*

For the purposes of my system, I will not use the entire infrastructure of PKI as detailed above because the keys will not be transmitted across any network, but will be locally stored. Therefore, I will use a simplified form of PKI instead, called SPKI. SPKI does not deal with public authentication of public key information and does not associate users with persons. Lastly, third party for authorization is not required[17].

2.3 What is a Cryptographic Hash Function?

A cryptographic hash function turns plaintext into cyphertext, and it is very hard to invert[18]. A hash function is any function that is used to map data of any arbitrary size to data of fixed size.

Given a cryptographic hash function $h(x)$, it must provide[18]:

- Compression –output length is small
- Efficiency – $h(x)$ easy to compute for any input data
- One-way –given a value y making it impracticable to find an x such that $h(x)$ equals y
- Weak collision resistance –given x and $h(x)$, making it impracticable to find y does not equal x such that $h(y)$ equals $h(x)$
- Robust collision resistance –making it impracticable to find any x and y , with x not equaling y such that $h(x)$ equals $h(y)$

Hash functions are very useful in security. In one example, someone can sign a message M by using their private key to encrypt the message and send it over to the recipient, and the recipient can verify the signature. However, if M is large, encrypting the message can be costly to compute, therefore, a cryptographic hash function would be ideal. There is a cryptographic hash function h , and $h(M)$ can be viewed as the “fingerprint” of the message. $h(M)$ is much smaller than the original message, but it identifies M .

Chapter 3

Related Work

Though my system is the first of its kind, it has surely had its spheres of influence. Most notably is Google Analytics (GA) which is widely used. The number of websites that uses GA is vast, and a website owner's objectives for using an analytical device are numerous. Though the number of objectives are endless, they can be boiled down to five generic categories[8]. These categories are:

- e-commerce
- lead generation
- content publishing
- online information
- branding

For e-commerce, one's objective is to use their website to sell goods and/or services for profit. By using an analytical device, one is able to study user activity, to create target advertisements, and then ultimately, to generate revenue. For lead generation, the goal is to collect user information to encourage user traffic and retention. For content publishing, the objective is to encourage visitors to return to the website and to see what it is that has them coming back. For online information, it is important for visitors to find what they are looking for when they need it, so the owner would be interested in what the visitor is and is not able to find. Lastly, for branding, the key is awareness and loyalty to the website.

FIGURE 3.1: Leakage of OSN ID to a Third-Party [3]

3.1 Personally Identifiable Information of a user being leaked to third-party servers

Aggarwal [3] tries to find out if online social networking sites (OSNs) are accumulating Personally Identifiable Information (PII) from the users on their websites. She analyzed several OSNs using *Live HTTP Headers* to study different parts of the HTTP headers. By analyzing these parts, she was able to determine that there were multiple types of “leakage” to these websites. The different types of leakage happen via Referer Header, Request-URI, and Cookie, and each of them are displayed in 3.1. She also realized that PII is leaked to third-party servers. Figure 3.2 shows two of such examples.

Since OSNs can divulge the PII of their users to anyone who is willing to pay for the data, one can fathom that nothing is stopping them from being able to share this data with the primary organizations/people my proof of concept is trying to safeguard the data from, government agencies. This is what my thesis tries to solve by not having this data readily available. The tracking data of the users is not divulged to third-party servers. Moreover, even if the Federal Bureau of Investigation (FBI) asked for the PII of all of the visitors of the website, the website owner could only provide the one-way hashed version of the visitors’ PII, which is meaningless to the FBI. If the FBI asked for the decrypted version, there

FIGURE 3.2: Leakage of Pieces of PII to a Third-Party[3]

would be no way to attain that version since it is impossible to invert data that has been one-way hashed.

3.2 UsaProxy

Atterer, Wnuk, and Schmidt [4] investigate how to track user activity using standard web technologies. They create a non-intrusive tracking tool. The system architecture is depicted in Figure 3.3. Their tool intercepts all traffic and outputs the data to a log file and track users' activity, such as mouse movements and keyboard input, in detail through client-side code written in JavaScript (JS), while remaining unnoticeable to the user. Researchers were trying to track what was the most frequently used path through a website, average time spent on a page, and mouse position. The most important part of their proxy, UsaProxy, is ultimately logging the user activity. A sample log file is depicted in Figure 3.5.

After UsaProxy was implemented successfully, the researchers analyze the usefulness of the approach. They conduct the test on 12 test participants who use a computer whose web browser uses the UsaProxy instead of connecting to the websites directly. The users are told to perform different tasks on two different websites. An example of mouse trails recorded by the HTTP proxy is displayed

FIGURE 3.3: System Architecture.[4]

in Figure 3.4. Afterwards, log files are produced from the proxy during the test. A snippet of a log file is displayed in Figure 3.5. The events in the figure show mouse movements, mouse overs, the cursor moving into an input field, and other similar activity.

In this example, the UsaProxy does a sound job in logging the user activity, but if an attacker were to get a hold of the files which does not seem like an arduous task, they would be able to understand what the files mean judging from the clarity of the log files. This is what my system tries to prevent: easy access to the log files to preserve the integrity of the website owner using the analytical system.

3.3 User Interest Based on Mouse Movement

Mueller and Lockerd [19] detail their work on how to take into account all mouse movements on a web page as another way to infer user interest. Their idea is to use scripting embedded into the websites to track mouse movement (position and their timestamps) and to send the logged data to a server for further analysis. The approach is limited to tracking scrolling and keypresses, but individual objects

FIGURE 3.4: Mouse trails recorded by the HTTP proxy[4]

```

141.84.8.77 2005-10-25,11:5:57 http://www.kiko.com/ serverdata 12
141.84.8.77 2005-10-25,11:5:58 http://www.kiko.com/ load width=1280;height=867
141.84.8.77 2005-10-25,11:6:2 http://www.kiko.com/ mousemove x=672;y=7
141.84.8.77 2005-10-25,11:6:2 http://www.kiko.com/ mouseover x=731;y=457 target=link:http://www.kiko.com/contact.htm+linktext:Contact
141.84.8.77 2005-10-25,11:6:6 http://www.kiko.com/ click x=815;y=231 target=id:SPAN16
141.84.8.77 2005-10-25,11:6:37 http://www.kiko.com/app.htm?use_auth=678397351 mousemove x=849;y=352
141.84.8.77 2005-10-25,11:6:37 http://www.kiko.com/app.htm?use_auth=678397351 mouseover x=472;y=296 target=id:DIV144
141.84.8.77 2005-10-25,11:6:37 http://www.kiko.com/app.htm?use_auth=678397351 mouseover x=161;y=229 target=id:left_bar
141.84.8.77 2005-10-25,11:6:38 http://www.kiko.com/app.htm?use_auth=678397351 click x=147;y=183 target=unknown:scrollbar
141.84.8.77 2005-10-25,11:6:40 http://www.kiko.com/app.htm?use_auth=678397351 mousemove x=148;y=138
141.84.8.77 2005-10-25,11:6:50 http://www.kiko.com/app.htm?use_auth=678397351 click x=26;y=507 target=id:IMG14
141.84.8.77 2005-10-25,11:6:50 http://www.kiko.com/app.htm?use_auth=678397351 focus
141.84.8.77 2005-10-25,11:6:56 http://www.kiko.com/app.htm?use_auth=678397351 keypress key=T
141.84.8.77 2005-10-25,11:6:56 http://www.kiko.com/app.htm?use_auth=678397351 keypress key=e
141.84.8.77 2005-10-25,11:6:56 http://www.kiko.com/app.htm?use_auth=678397351 keypress key=s
141.84.8.77 2005-10-25,11:47:45 http://de.wikipedia.org/wiki/Hauptseite scrolledTo y=399

```

FIGURE 3.5: Small sample of the log output produced by the UsaProxy.[4]

on the web pages, such as buttons are not recognized by the client-side code. Furthermore, their system posts the accumulated data with embedded scripting, and the data is analyzed and stored on a server. Their collection technique does not require any additional software on the user's browser.

3.4 Unobtrusively Studying User's Behavior

Goecks and Shavlik [20] talk about unobtrusively studying users' behavior. They research actions such as scrolling and bookmarking webpages to determine user interest. Therefore, when the user navigates to a new page, these are recorded: the text of the HTML text, the number of hyperlinks the user clicked on, the amount of user scrolling activity, and the amount of user mouse activity.

3.5 My Work

After perusing numerous peer-revised journals, including a few that have been summarized in this thesis, it has become quite evident that the researchers were more interested in tracking user activity and less concerned with securing the tracking data. The researchers devise unique ways of tracking user activity and storing the data in a log file, but there is little focus on encrypting the tracking data so prying eyes cannot decipher exactly what the user did. Therefore, if an attacker manages to get a hold of the log files, s/he would be able to make sense of the data since it was all in plaintext. That would not be the case for my system since the data is encrypted with a constantly expiring key and PII is hashed, which makes it impossible to convert back to plaintext. What my thesis tries to emphasize is the importance of keeping the tracking data and the users' PII away from prying eyes. The purpose of my thesis is to create a proof of concept of a data analytical device that securely stores the tracking data in a database. It records the actions that the user took on the particular website and encrypts the data that is sent from the NodeJS server to the database before being stored. Also, the tracking data of the user is encrypted with a symmetric key that is then encrypted with a public key. The symmetric key pair expires after a week and regenerates a new pair after the expiration date, giving an attacker or a government agency a small timeframe to decrypt the data should they get a hold of the private key.

By using my system, a website owner is guaranteed multiple things. For one, all of his visitors' PII is kept confidential and is unlikely of being discovered. Second, all of the visitors' tracking data is encrypted before being stored in the database. The symmetric key that encrypts the data is encrypted with a public key, and the symmetric key pair expires over a short period of time, so that even if an attacker gets a hold of the private key, s/he must need to find the symmetric key that encrypted the data. By that time, s/he has a limited time in order to decrypt the data. Therefore, if time runs out before s/he can decrypt the data, the analytics site's half of the symmetric key pair will be encrypted with the public key, and it would be the attacker's job to hunt for the new symmetric key again.

With these security measures in place, a website owner's visitors can feel comfortable knowing that their tracking data is safeguarded, except for a short timeframe of being decrypted by hackers or government agencies should they get a hold of the private and/or symmetric key. Moreover, website owners can encourage more visitors to their website by letting them know that their information is safeguarded from prying eyes. For the website owner, s/he can continue to view the data since his/her private key does not expire. To summarize, the biggest benefit is knowing that government agencies cannot get a hold of the data and even if they do, they cannot decrypt the PII of the users as it has been one-way hashed nor the tracking data as a new symmetric key pair expires too often should they get a hold of the website owner's (or WO for short) private key.

Chapter 4

Design and Architecture

From Day 1 of implementing the system until the end of the week, visitors will visit the WO's website and perform some activity. This tracking data (TD_n in Figure 4.1) is sent to the analytics site that will be encrypted with a symmetric key to produce ciphertext. The symmetric key is then encrypted with a public key. The symmetric key that was used to encrypt the tracking data from Day 1 till the end of the one week period is called K_0 , and every subsequent key will be K_n , where n is the number of weeks since the system's inception. All of the encrypted analytics data (C_n in Figure 4.1) will be stored in a database along with a corresponding ID (ID_{K_n} in Figure 4.1) pertaining to the symmetric key that was used to encrypt the tracking data. This way, if the website owner decides that they want to view the data for a particular visitor, they would know the symmetric key that was used to encrypt that visitor's tracking data by the corresponding ID. That way, s/he will know which symmetric key will be used to decrypt the analytical data. The symmetric key that encrypts the data for that time period is encrypted with the WO's public key and stored in a database. This way, if a FBI agent demanded that the analytics site produce the tracking data, all they would be able to provide is meaningless ciphertext that was encrypted with a symmetric key, which is encrypted with a public key. In order to decrypt the symmetric key, the FBI would need to get a hold of the private key. The private key belongs with the WO, who is out of jurisdiction. My system generates the public and private keys and transmits the private key to the website owner to enable access to the analytical results via an encrypted email sent upon validated registration to the system.

FIGURE 4.1: Encrypting Analytics Data

The WO requests the analytics site to generate a report when s/he wants to see the analytical data that has been generated over a period of time. The analytics site will send back the encrypted analytical data, publicly encrypted symmetric keys that were used to encrypt the data, and a KeyMap. This KeyMap maps ciphertext ID to its corresponding key ID. This key ID allows the WO to know which symmetric key was used to encrypt the desired ciphertext. Once the WO knows which symmetric key was used to encrypt the tracking data, he could use his private key to decrypt the corresponding symmetric key. Once he has a hold of the symmetric key, he can easily decrypt the ciphertext to produce the plaintext analytical data, thus generating the report as depicted in Figure 4.2.

In Figure 4.2, K_0 and K_1 correspond to the symmetric keys that were used to encrypt the analytical data. K_0 corresponds to the symmetric key that was used to encrypt the data for the first week and K_1 corresponds to the second week. K_n would correspond to the $n+1$ week that the system has been in existence. $\{K_n\}_{WSO}$ corresponds to the symmetric key after it was encrypted with the public key corresponding to the $n+1$ week since the system's inception. ID_{K0} corresponds to the ID of the symmetric key that was used to encrypt the analytical data. C_0 is the ciphertext or analytical data after it was encrypted with its corresponding symmetric key, K_0 . ID_{C0} is the ID of the corresponding ciphertext. Once the symmetric keys have been decrypted, they become K_n' , which are then used to

FIGURE 4.2: Generating Report

decrypt the analytical data to produce P_n , the plaintext analytical data that is human readable.

The KeyMap looks like this:

cyphertextID	keyID
C_{ID0}	ID_{K0}
C_{ID1}	ID_{K1}
C_{ID2}	ID_{K2}

Chapter 5

Implementation

In order to create my system, I used the MEAN stack. MEAN is an acronym that stands for:

- MongoDB
- ExpressJS
- AngularJS
- NodeJS

5.1 MongoDB

MongoDB is a NoSQL, document-oriented database [\[21\]](#). Its purpose in the thesis is to store the tracking data and the PII of the users. It has both a community edition and an enterprise edition. The enterprise edition offers an option to SSL-encrypt the pathway from the server to itself preventing attackers from snooping on the data being sent, which I chose to go with when setting up my system.

I chose MongoDB over other databases like MySQL and Cassandra because MySQL does not use the SSL (Secure Sockets Layer) protocol for secure connections between the client and server, though it does support TLS (Transport Layer Security), the successor to SSL. Secondly, Cassandra does support the SSL protocol,

but I find establishing the SSL connection between the server and database significantly easier for MongoDB as opposed to Cassandra. Thus, I chose to go with MongoDB as the systems database.

5.2 ExpressJS

ExpressJS is a Node.js framework, which is used to build single-page, multi-page, and hybrid web applications[22]. In the context of the thesis, its purpose is to do all of the sophisticated work of matching specific HTTP requests to code that needs to be executed to service the request.

5.3 AngularJS

AngularJS is an open-source web application framework developed by Google[23]. It was developed to address the challenges encountered in developing single-page applications. It is the frontend portion of the MEAN stack. Essentially, it is executed on the browser and handles the button pushing that takes place on my system.

There are at least two other MV* frameworks I could have gone with, but I chose to go with AngularJS. Those other two frameworks are Backbone and Ember[6]. MV* stands for Model-View-Wildcard, where the Wildcard can refer to C for Controller, VM for View-Model, P for Presenter, and there are many others[24]. One of the more popular design patterns, MVC (Model-View-Controller), was designed to separate the user interface logic from the business logic[25].

1. **Model.** The model manages the data of the application domain, and responds to the View for information about its state, and to the Controller for instructions to change state.
2. **View.** The view manages the display of information.
3. **Controller.** The controller interprets the user's mouse and keyboard inputs, informing the model and/or the view to change as appropriate.

FIGURE 5.1: MVC[5]

Figure 5.1 provides a better illustration of MVC and how each component depends on the others.

AngularJS has the largest community out of the two other frameworks, which means that the framework is stable and robust. There is a lot of support and features offered by AngularJS with a large Question and Answer section on the popular developer website, Stack Overflow. When a significant number of people are working on an open-source framework, they will be able to recognize any bugs that occur in the framework as opposed to a framework that is not as widely used.

Figure 5.2 illustrates the popularity of AngularJS compared to the two other similar frameworks.

Though Figure 5.2 shows only the current state of each framework, Figure 5.3 shows the popularity growth of each of the frameworks with AngularJS having a significantly higher growth rate compared to the other frameworks.

Also, page load times are crucial for the success of the website, since the user will not exhibit much patience if the application takes a while to load. When looking at Figure 5.4, considering the size of the framework is not enough (the second column), one must also include the size of the framework's corresponding dependencies (the third column) to determine the actual size of the framework to

Metric	AngularJS	Backbone.js	Ember.js
Stars on Github	40.2k	18.8k	14.1k
Third-Party Modules	1488 <i>ngmodules</i>	256 <i>backplugs</i>	1155 <i>emberaddons</i>
StackOverflow Questions	104k	18.2k	15.7k
YouTube Results	~93k	~10.6k	~9.1k
GitHub Contributors	96	265	501
Chrome Extension Users	275k	15.6k	66k
Open Issues	922	13	413
Closed Issues	5,520	2,062	3,350

FIGURE 5.2: Comparison of AngularJS' Popularity to Other Frameworks'[6]

FIGURE 5.3: Comparison of AngularJS' Popularity Growth Rate to Other Frameworks'[6]

be loaded. Therefore, out of the three frameworks, AngularJS is the smallest size to load with only 39.5kb compared to Backbone's 64.1kb and Ember's 136.2kb.

Lastly and most importantly, AngularJS has received significant corporate backing, since it was developed by Google. Therefore, with so many advantages to using AngularJS, it was an easy decision to go with AngularJS.

Framework	Net Size	Size with required dependencies
AngularJS 1.2.22	39.5kb	39.5kb
Backbone.js 1.1.2	6.5kb	43.5kb (jQuery + Underscore) 20.6kb (Zepto + Underscore)
Ember.js 1.6.1	90kb	136.2kb (jQuery + Handlebars)

FIGURE 5.4: Comparison of AngularJS' Size to Other Frameworks' [6]

5.4 NodeJS

Node.js is an open-source, cross-platform runtime environment for developing server-side Web applications[26]. Node.js is not a JavaScript framework, but its applications are written in JavaScript. It served as the server in my thesis and handled requests generated by the client. It would also send encrypted data to the database over SSL.

5.5 A Further Look Into The Architecture

The security precautions that my system takes are as follows. It uses the open-source web server called Nginx, but in this case, it creates a reverse proxy server to establish a HTTPS (Hyper Text Transfer Protocol Secure) connection from the browser to the server. In insecure connections between the browser and the server, an HTTP connection is created. The difference between HTTPS and HTTP is that with HTTPS the data sent from the browser to the server is encrypted using SSL.

When one visits an SSL-encrypted website, s/he would see what is depicted in Figure 5.5 in the address bar.

Not only is the pathway from the browser to the server SSL-encrypted, the pathway from the application server to the database is SSL-encrypted as well. SSL is a security protocol that allows sensitive information like credit card information and

FIGURE 5.5: HTTPS in the Browser^[7]

social security numbers to be transmitted securely. The SSL protocol determines how the necessary information is encrypted and sent to the corresponding recipient. All browsers have the capability to interact with secured web servers using the SSL protocol. When the user registers at my website, as mentioned previously, public and private keys are generated using Diffie-Hellman encryption functions in NodeJS. These keys are always within the memory of the local server and are not transported across any network. Furthermore, it is important that the NodeJS binary code bindings be obtained from a trusted and authenticated source, so that there is no opportunity for a hacker to compromise the NodeJS binaries by

embedding malware into the binding.

In the system, keys are generated by calling `crypto.getDiffieHellman(groupName)`. This creates a predefined DiffieHellman key exchange object. ‘Modp14’ is the supported group. The returned object mimics the interface of objects created by `crypto.createDiffieHellman()`, but this will not allow changing the keys (with `diffieHellman.setPublicKey()` for example) [27]. This functions advantage is that the parties do not have to generate nor exchange a group modulus beforehand, saving both processor and communication time. Furthermore, this enables security protection because the group modulus is not being exchanged across processes. The keys returned are 256 bytes long (2048 bits) each. These keys are sufficiently long because they reduce the probability of a hacker randomly guessing the right combination of bits to a chance of $1/(2^{2048})$. The private key is sent to the WO via an encrypted email sent upon validated registration to the system. As mentioned previously, the symmetric key pair is changed on a weekly basis in order to reduce the probability of a successful hack because the target has moved while the hacker is still preparing to attack it. The way to implement this is to use a NodeJS `setTimeout()` function with a callback parameter that will be invoked when the timeout expires. The `setTimeout()` function requires a timeout value in milliseconds. Therefore, for a weeks duration, the timeout value should be set to $3600 \text{ seconds} * 10^3 \text{ milliseconds} * 24 \text{ hours} * 7 \text{ days} = 604,800,000 \text{ milliseconds}$. This is well within the maximum timeout limit of approximately 25 days [28]. Therefore, there should be no problem implementing it for a 7-day timeout.

Furthermore, this encrypted email will enforce that a return acknowledgement email be sent back to my system to ensure that the encryption credentials were opened by an authorized person on the website owner’s email domain, not any unauthorized third-party. The email headers of the returned acknowledgement will be examined by my system to ensure that it was opened at the properly authorized IP address. Furthermore, overall security is additionally enhanced by enforcing a twelve-hour time window for the credentials to be opened by the authorized, registered website owner. If a properly authenticated returned email acknowledgement is not received within this twelve-hour time window, those keys are invalidated, a new pair of keys is generated, and a new email is sent. This process will repeat until a properly authenticated returned email acknowledgment is received within the twelve-hour time window.

The image shows a web interface for user registration and login. At the top, there are two links: "Register" and "Login". Below these, there are two input fields: "Username" with the value "john" and "Password" with masked characters ".....". A green "Register" button is positioned below the password field.

FIGURE 5.6: Login Interface

The image shows a web interface for user activity. At the top, there are two links: "Logout" and "Signed in as john". Below these, there is a large heading "Tracking Data" and a text box containing the hexadecimal string "5692f458829ada25062ae46f 29".

FIGURE 5.7: Activity Interface

The username is encrypted and their corresponding passwords are one-way hashed. Figure 5.6 shows what the new user sees when they are attempting to register at the website.

If they are successfully able to register at the website, they are directed to the homepage where they click “play” on a video in a separate link and the counter for the number of clicks will increase and be outputted. Also, the id for the video being clicked is encrypted to further bolster the security of the website. Figure 5.7 illustrates what the user sees when they are able to successfully register or login to the website.

By cryptographically hashing the passwords, it makes it impossible to invert. That means that the plaintext password is converted into a meaningless string, but that meaningless string cannot be converted back into the plaintext password.

```
MongoDB Enterprise > db.trackingdatas.find()
{ "_id" : ObjectId("5692f458829ada25062ae46f"), "_ct" : BinData(0,"JnXSR2y8bBzo1Czwz0ZajMCtMe38W3tKDTo1s
qoaYOM="), "username_": "john", "__v" : 0 }
```

FIGURE 5.8: Encrypted Tracking Data

This makes it impossible for the user to understand what his password means after being hashed, let alone the subject of the thesis, a government agency. By doing so, none of the users' PII can be exposed alleviating users' concerns of the consequences of having their PII compromised.

For a properly authorized and authenticated website owner to access my system, that person will access my secure website via HTTPS server, open a SSL-secure session with the proper username and password, and enter the key information to access the results stored in my system. Any improper login or key information will be deemed to be an unauthorized attack and cause the website to be locked out for a period of one hour. Furthermore, the current keys will be encrypted with WO's public key and the new symmetric key is used for future analysis. A new set of keys will be emailed as per the procedure described above. This safeguard will prevent automated bot attacks from trying to speedily discover the appropriate keys. Additionally, the originating IP address of the malware attack will be logged, blacklisted, and excluded from my system server's domain response.

5.6 What a Hacker Would See

If a hacker seizes tracking data being stored in the database, s/he will not be able to make any sense of it. Figure 5.8 displays what the hacker would see if s/he were to hack the system and get a hold of the tracking data.

The only information that the user would have is the first name of the user who signed up on the website. The actual tracking data is encrypted.

If the hacker were to get a hold of the user data, s/he would also only have the name of the user but not his/her corresponding password as that has been one-way hashed as can be seen in Figure 5.9.

Knowing this, a user should feel confident about signing up for the particular website implementing the analytical system as I have gone to great lengths to

```
{ "_id" : ObjectId("569de883a95c7c904892bdc"), "password" : "$2a$10$rW66mMzD5XnSmlo8MMLfW0HNbXP3MZfJKfh
1.bCzXM0B7wQX47ADS", "username" : "john", "privateKey" : { "key" : "00096a8d988fff36af9dfe5ef92ee45042c2
a8f65ca56e87a1a2b879bc70c9e86f8cbcc2b34ba9a92419cd970497187dbdafbec72663d10ec40787d2c19be83b466b221a31a3
c5956dc3340d1c1f4bb40e7dadad6470e2135b9ae624e868bf76145c6aa18a02e435c43cabe3f158bf5d004aba6d34a7fc7c6780
bcd919809fce3e6e88f38b7af40cdde7dcce34488e1540361912a2565c902d334b3e60e05afe8c2efaed1421872e4cba2cd0019b
a66327e1669de9b4b08e024a7957552302bc9ffc2093c70da6ddae9a46cd81aedb5ae9c53508879dc175f0c578f92250c245087f
ac734ec8e115b056f23eb261120fea981427080b996906c47d2d16d967ab", "timestamp" : ISODate("2016-01-19T07:40:5
5.232Z") }, "__v" : 0 }
```

FIGURE 5.9: Encrypted User Data

keep the critical data secure and unreadable to hackers who manage to get a hold of the data.

5.7 How a Website Owner Accesses Data

The website owner can only access encrypted data via a NodeJS script that can only be accessed through secure shell (SSH). No world-accessible web front-end is provided, which limits the number of potential vulnerabilities that can be exploited to access sensitive data. Another away of putting this is that the website owner can only access the encrypted data through SSH, which is a technology that is mature and known to be highly secure. Building a custom web front-end that is accessible to the Internet would risk introducing a number of vulnerabilities that could be exploited. The script itself uses the same technology that the website back-end server uses. The MongoDB database is accessed through NodeJS via Mongoose, an elegant MongoDB object model for NodeJS. Mongoose is aware of the encryption that has been performed on the MongoDB documents, so it is automatically decrypted when queried from the database. The unencrypted form of the data only exists in the server's memory for the short period of time while the script is being executed. When the website owner wants to view the user and tracking data, s/he would use their private key to decrypt the symmetric key to finally decrypt the tracking data. A portion of the decrypted data is the user's name and the tracking data as depicted in Figure 5.10

The website owner could also decrypt the data stored in the “user” table, and the data is displayed in Figure 6.2. As one can see, the user's name and additional information is provided. What is not provided is the user's password as that has been one-way hashed corroborating the fact that when data has been one-way

```

Justins-MacBook-Pro:jda6WithoutComments Justin$ node extractData.js
mongodb connected
[ { count: 29,
  __v: 0,
  username: 'john',
  _id: 5692f458829ada25062ae46f } ]
_

```

FIGURE 5.10: Decrypted Tracking Data

```

e9f2a' },
  __v: 0,
  username: 'john',
  _id: 569dec4c9beeca5a0575e332 },
  { oldPrivateKey: {},
    privateKey:
      { timestamp: Mon Jan 18 2016 23:57:53 GMT-0800 (PST),
        key: '95388db4aec6931d94e92d70e7746246d4c9a27b7fa354c5bf275e942be6a8fb4c15f5187e27564e1879c967afd
2338274f9d5cafbc0f17bc07b32f8e74507f629906903a5cde5e7a355f3a572e6a498f957b030b595482129edf59ce6d1bc0ee4
3b13060383c1a53ee333fe1b4e41230c1fd004fe5eff86a53276f85a5acd7984eb66a484e1ae73b99cd596893bee4a6ac5d48e98
5d0d574bdf256c3e08e02fc6ad89614d2f83d58d5f70a31aef07f64ca6c425362f9d62fb007f4072cc38bfe731a854d713907457
ee362f6ff2f21580ae62e275d1af0d1a962b8032a1f2a998adea3797869cfcf47391a5786b709e048ffed9fef2afdeba4bd56249
6d986' },
    v: 0.

```

FIGURE 5.11: Decrypted User Data

hashed, there is no way to get the unencrypted form of that data further bolstering the security of the system.

5.8 SSH

When developing a web application, all relevant code exists on the development machine. In this case, it is my laptop. When a web application is actually launched into production, however, the code is put on a server in some remote location. This server runs all the programs that were run on the development machine during production (e.g. Nginx, MongoDB, etc.), but is configured in such a way that any machine on the Internet has access to the web application. This contrasts to what was done while developing, during which time the web application was accessible only on the development machine.

In order to set up the remote server (i.e. install all necessary programs, write the configuration scripts, put code files in the appropriate locations), one uses SSH. SSH stands for “Secure Shell” and is an industry standard way of accessing remote machines securely. By using this technology to access the encrypted data generated by users (as described above), one is guaranteed a much more secure

FIGURE 5.12: System diagram

system than if they were to write a web front-end to access the secure data. SSH is simple and proven, whereas a web application administration system involves a number of additional technologies (NodeJS, AngularJS, Nginx), each of which can introduce vulnerabilities in the system.

5.9 Bringing It All Together

After discussing the multiple parts of the system, it's time to bring it all together. Figure 5.12 is a diagram of the entire system from when the user tries to log in to the website, to how the data is sent from the web server to the application server, and how the data is sent from the application server to the database to finally be stored at rest. All the while, the three pathways are SSL-encrypted.

Chapter 6

Experiments

In order to verify my system runs according to expectations, I created some test cases. I wanted to test whether my system encrypts and decrypts the tracking data, how long does it take to generate the report, and what factors influence the length of time it takes to generate the report.

6.1 Does tracking data get encrypted?

The first case that I wanted to verify is whether or not the tracking data gets encrypted prior to being stored in the database. Figure 6.1 shows the tracking data for ten visitors to the website. All that is visible to an attacker are the names of the visitors, but what they did on the site is incomprehensible.

This proves that my system is able to send tracking data from the browser to the analytics site in a safe, secure manner, encrypt the data with a symmetric key, and have it be stored inside the database.

```
MongoDB Enterprise > db.trackingdata.find()
{ "_id" : ObjectId("91ea2D1673594e858fB4f40"), "_ct" : BinData(0, "1b272643D6F04F54BFD872FF23A80be8"), "username" : "mxna", "_v" : 0 }
{ "_id" : ObjectId("654BbC370a5E447190F4035"), "_ct" : BinData(0, "6cb54c26Da634D038Dab1A0564923152"), "username" : "scmu", "_v" : 0 }
{ "_id" : ObjectId("F059d1fceAe4d8288016b0"), "_ct" : BinData(0, "d88a451903154a5d8d0E9F7C958c750b"), "username" : "wnix", "_v" : 0 }
{ "_id" : ObjectId("3D9aa8015a57424C9D29b64"), "_ct" : BinData(0, "23ed6530401344FFA3E4F8968F00bc86"), "username" : "pqgm", "_v" : 0 }
{ "_id" : ObjectId("3070FA432bB1475ea7d50A9"), "_ct" : BinData(0, "E7FB392692F143B7B90F6301C3f7C14d"), "username" : "azqt", "_v" : 0 }
{ "_id" : ObjectId("b0e70509C51A42e58df4f54"), "_ct" : BinData(0, "0a770150F3CC4E6CBf0D1a5325726d53"), "username" : "vcno", "_v" : 0 }
{ "_id" : ObjectId("8D8A84D48D0347d0b59e033"), "_ct" : BinData(0, "Ce4978920c9d4Ed80a113b5f1352611"), "username" : "xoao", "_v" : 0 }
{ "_id" : ObjectId("3B98AD4b2395409Cb38EAB"), "_ct" : BinData(0, "87f63663137e48FF84960Ce9d9CD02Fb"), "username" : "wsnl", "_v" : 0 }
{ "_id" : ObjectId("8Ea502B9847d4C36b1C476"), "_ct" : BinData(0, "6B3EDa448e414C6797678e0615f8b7ea"), "username" : "ljgb", "_v" : 0 }
{ "_id" : ObjectId("e9d0DaeA61140468E8Aa67"), "_ct" : BinData(0, "3dca7E263E2D43e5B3bf37c2754f0116"), "username" : "naxt", "_v" : 0 }
```

FIGURE 6.1: Encrypted Tracking Data

```

Justins-MacBook-Pro:jda6WithoutComments Justin$ node extractData.js
mongodb connected
[ { count: 86
  __v: 0,
  username: 'mxns',
  _id: 91ea2D16735B4c858fB4f40 } ]
[ { count: 27
  __v: 0,
  username: 'scmu',
  _id: 654BbC370a5E447190F4035 } ]
[ { count: 35
  __v: 0,
  username: 'wnix',
  _id: FD59D1fceAAe4d8288016b0 } ]
[ { count: 76
  __v: 0,
  username: 'pqwm',
  _id: 3D9aa8D15a57424C9D29b64 } ]
[ { count: 9
  __v: 0,
  username: 'azqt',
  _id: 3070FA432bB1475cA7d50A9 } ]
[ { count: 13
  __v: 0,
  username: 'vcno',
  _id: b0e70509C51A42e58dF4f54 } ]
[ { count: 76
  __v: 0,
  username: 'xoao',
  _id: 8D8A84D48D0347d0b59e033 } ]
[ { count: 64
  __v: 0,
  username: 'wsni',
  _id: 3B98AD4b239540BCb38EEAB } ]
[ { count: 14
  __v: 0,
  username: 'ljgb',
  _id: 8Ee502E9847d4C36Bb1C476 } ]
[ { count: 80
  __v: 0,
  username: 'nsxt',
  _id: e9d0DaeA661140468E8Aa67 } ]

```

FIGURE 6.2: Decrypted Analytical Results

6.2 Is the tracking data capable of being decrypted?

The second case that I wanted to verify is whether or not the tracking data is capable of being decrypted for the WO's use. Figure 6.2 shows what the WO would see when they are able to decrypt the encrypted tracking data, showing the usernames of the sample of ten visitors and the number of times they clicked on the video on the website, respectively.

This proves that the website owners are able to request the analytical data from the analytics site upon proper authentication, receive it in its entirety, and be able to decrypt the data with their private keys in order to get the plaintext version of the tracking data.

6.3 How long does it take to generate the report?

The third case that I wanted to test for is how long does it take to generate the analytical report. Inside the report are the visitors' names, the number of times they clicked on the video, and their unique ID to distinguish between users, primarily in the case where they share the same name.

I generated reports for a site with 500 new visitors per week for 4 weeks:

- First week: 500 visitors, 1 key
- Second week: 1000 visitors, 2 keys
- Third week: 1500 visitors, 3 keys
- Fourth week: 2000 visitors, 4 keys

# of Visitors	Time for unencrypted (ms)	Time for encrypted (ms)
500	33.1	64.4
1000	63.1	126.1
1500	94.3	192.1
2000	124.0	257.0

TABLE 6.1: Milliseconds to Generate Report Depending on Number of Keys

Table 6.1 displays the number of milliseconds it takes to generate the report when data is unencrypted and encrypted prior to being stored in the database with the number of milliseconds steadily increasing as the number of visitors and the number of symmetric key pairs increase. This is to be expected since as the time increases between the system's inception and when the WO requests the generated report, more encrypted symmetric keys will be sent over to the WO who has to decrypt the data with more of his symmetric keys. This increase in resources needed to generate the report is what leads to an increase in the number of milliseconds needed to generate the report. Also, with more visitors' data to decrypt, the longer it will take to generate the report.

Figure 6.3 shows a graphical comparison of the time it takes to generate the report when the data being stored in the database is unencrypted versus when

FIGURE 6.3: # of Visitors vs Time to generate report in ms

it is encrypted. There is a linear increase in the time to generate the report in milliseconds with an increase in the number of visitors for unencrypted and encrypted tracking data. Also, the time to generate the report for encrypted data is greater than the time for unencrypted data for each of the number of visitors (500, 1000, 15000, and 2000), since there is the added overhead of decrypting the encrypted data when generating the report, which is not required for the already unencrypted data. Therefore, the encryption algorithm scales well.

Chapter 7

Discussion

My system encrypts the tracking data with a different symmetric key that expires after a certain period of time. After that period has passed, that particular symmetric key is encrypted with my system's public key, is stored in a database, and a new symmetric key pair is created for the next week. Therefore, if the FBI came knocking on my door on Week 5, where Week 1 is the first week after the system was first implemented by the website owner, the FBI would have no way of being able to access the data that was accumulated from Week 1 to 4. This is because my system encrypts each of the symmetric keys that were used to encrypt the tracking data with the website owner's public key every week. The FBI would need to get a hold of the private key in order to decrypt the symmetric keys, which is with the website owner. Since the website owner is the only one who has access to the private key, he can use his private key to decrypt each of the symmetric keys that was encrypted with the public key. Now that the symmetric key has been decrypted, he can use that symmetric key to decrypt the tracking data. The novelty is that if the FBI comes knocking on the system owner's door on day 3 of a 7-day week, the system owner would only be able to divulge the analytical data for that week. Therefore, if the system were implemented for 40 weeks so far, the system owner would only have to be able to divulge 3 days worth of data as opposed to 280 days. This bolsters the website owner's confidentiality. It bolsters the owner's integrity in that it maintains the consistency, accuracy, and trustworthiness of data over its entire life cycle[29]. My system ensures the site's integrity by SSL-encrypting the pathways from the browser to the server and from the server to the database, preventing attackers from snooping on the data in transit.

This scenario of safeguarding data is very closely related to a recent feud between the tech giant, Apple Inc., and the FBI. In December 2015, there was a terrorist attack carried out in San Bernadino, California by Syed Farook and his wife. The investigation by the FBI has been ongoing for months and has asked that Apple create a new version of the iPhone operating system, circumventing several security features, and install it on an iPhone recovered from the investigation, which was being used by the terrorists. Though the FBI has maintained that they want this new operating system for this phone only, by creating this iOS that bypasses these security features, Apple has argued that it creates a backdoor. In essence, Apple would be generating a “master” key that could be used over and over again, on any number of devices, jeopardizing the privacy of Apple’s users.

My system’s main objective is to track user activity but to be able to store and safeguard this information from hackers and government agencies like the FBI. When visitors come to the WO’s website and perform activity, this tracking data is being sent securely to the analytics site via SSL. When the analytical data reaches the analytics site, it is encrypted with a symmetric key. When the expiration date is up, the symmetric key is encrypted with a public key and a new symmetric key pair is generated for the following week. The symmetric key along with its corresponding ID is stored in its own database, called the symmetric key database. Also, the data that is encrypted is stored in its own database along with its own ID to distinguish it plus the same ID that was stored in the symmetric key database, called the encrypted analytics data database. This is all the information that is available to the FBI if they were to demand the data. Therefore, it would be impossible to compromise the data that was gathered, since the encrypted analytical data is incomprehensible as is. If the FBI needed to read it, they would have to get a hold of the symmetric key in order to decrypt it. The only problem is that the symmetric key is encrypted with a public key that would need its private key counterpart in order to decrypt the symmetric key. The private key is with the WO who is out of jurisdiction. Therefore, the way the system is setup, it would not be possible for a FBI agent to compromise the data, bolstering the security, confidentiality, and integrity of the system and the website owners that implement it.

Chapter 8

Conclusion

I demonstrated how analytics data can be stored so that only the site owners can view the data about their customers. With my design, even the analytics site itself cannot decrypt the data after a given window of time has elapsed. The novel aspect of my design is that the analytical data can be securely stored and be protected against a government agency forcing the analytical site to release the data of all users. Tracking data is encrypted with a symmetric key that is then encrypted with a public key, and the symmetric key pair expires after a short period of time prior to being stored in the database. After that window has elapsed, the symmetric key is no longer accessible. Only the website owners have access to the private key for a lengthy period of time that allows them to go back and view the data.

8.1 Future Work

One of the things that my thesis did not address is “blindly” storing the keys with the server such that the server is unaware of how many keys were stored and their lengths. For my future work, I propose that the user keys be stored with the tracking data in a secure, encrypted database using MongoDB. This is done through Dynamic Searchable Symmetric Encryption (dynamic SSE) that allows a client to store a dynamic collection of encrypted documents with a server, and later quickly carry out keyword searches on these encrypted documents, while revealing minimal information to the server. When building the dynamic SSE scheme, “blind storage” is introduced, which allows a client to store a set of files

on a remote server in such a way that the server does not learn how many files are stored, or the lengths of the individual files. The approach in Reza Curtmola's work titled "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions" formed the basis for my work[\[30\]](#).

Bibliography

- [1] Public key infrastructure. (accessed February 14, 2016). URL <http://www.omnisecu.com/images/security/public-key-infrastructure/symmetric-encryption.gif>.
- [2] Intro to cryptography. 2010. URL http://www.infosectoday.com/Articles/Intro_to_Cryptography/CryptoFig05a.jpg.
- [3] Anupama Aggarwal. Summary of on the leakage of personally identifiable information via online social networks. August 2012. URL http://precog.iiitd.edu.in/teaching/CSE648_fall2012/files/Anupama_Leakage_of_PII_via_OSN.pdf.
- [4] Richard Atterer, Monika Wnuk, and Albrecht Schmidt. Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the 15th International Conference on World Wide Web, WWW '06*, pages 203–212, New York, NY, USA, 2006. ACM. ISBN 1-59593-323-9. doi: 10.1145/1135777.1135811. URL <http://doi.acm.org/10.1145/1135777.1135811>.
- [5] Mvc. 2016 (accessed February 12, 2016). URL <https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/2000px-MVC-Process.svg.png>.
- [6] Uri Shaked. Angularjs vs. backbone.js vs. ember.js. 2016. URL <https://www.airpair.com/js/javascript-framework-comparison>.
- [7] What Is SSL (Secure Sockets Layer) and What Are SSL Certificates? 2016 (accessed February 10, 2016). URL <https://www.digicert.com/ssl.htm>.
- [8] Analytics. (accessed February 22, 2016). URL <https://www.google.com/analytics/>.

- [9] Matt McGee. How many websites use google analytics? at least 10 million, google says. April 2012. URL <http://marketingland.com/google-analytics-is-installed-on-more-than-10-million-websites-9935>.
- [10] Ashley madison. 2016 (accessed on March 3, 2016). URL <https://www.ashleymadison.com/>.
- [11] These companies lost your data in 2015's biggest hacks, breaches — zdnet. 2015. URL <http://www.zdnet.com/pictures/biggest-hacks-security-data-breaches-2015/>.
- [12] Anthony Cuthbertson. Isis hackers join forces with pro-palestinian hackers anoghost. January 2016. URL <http://www.newsweek.com/isis-hackers-join-forces-pro-palestine-hackers-anoghost-413999>.
- [13] Bruce Schneier. Attacking tor: how the nsa targets users' online anonymity. October 2013. URL http://cyber-peace.org/wp-content/uploads/2013/06/Attacking-Tor_-how-the-NSA-targets-users-online-anonymity_-_World-news_-_theguardian.pdf.
- [14] encryption. (accessed February 27, 2016). URL <http://www.webopedia.com/TERM/E/encryption.html>.
- [15] public-key encryption. (accessed February 25, 2016). URL http://www.webopedia.com/TERM/P/public_key_cryptography.html.
- [16] Steve Lander. Advantages disadvantages of symmetric key encryption. (accessed March 1, 2016). URL <http://science.opposingviews.com/advantages-disadvantages-symmetric-key-encryption-2609.html>.
- [17] et al. C. Ellison. Spki certificate theory. pages 5–6, September 1999. URL <https://www.ietf.org/rfc/rfc2693.txt>.
- [18] Mark Stamp. Information security: Principles and practice. 2:126–127, 2011.
- [19] Florian Mueller and Andrea Lockerd. Cheese: Tracking mouse movement activity on websites, a tool for user modeling. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, pages 279–280, New York, NY, USA, 2001. ACM. ISBN 1-58113-340-5. doi: 10.1145/634067.634233. URL <http://doi.acm.org/10.1145/634067.634233>.

- [20] Jeremy Goecks and Jude Shavlik. Learning users' interests by unobtrusively observing their normal behavior. In *Proceedings of the 5th International Conference on Intelligent User Interfaces, IUI '00*, pages 129–132, New York, NY, USA, 2000. ACM. ISBN 1-58113-134-8. doi: 10.1145/325737.325806. URL <http://doi.acm.org/10.1145/325737.325806>.
- [21] MongoDB 3.2. 2016. URL <https://www.mongodb.org/>.
- [22] Express - node.js web application framework. 2016. URL <http://expressjs.com/>.
- [23] Angularjs by google. 2016. URL <https://angularjs.org/>.
- [24] Menon Jats. Mvc. 2016 (accessed February 12, 2016). URL <http://menonjats.com/>.
- [25] Model-view-controller. 2016 (accessed February 12, 2016). URL <https://msdn.microsoft.com/en-us/library/ff649643.aspx>.
- [26] Node.js. 2016. URL <https://nodejs.org/en/>.
- [27] Node.js v6.0.0 documentation - crypto. 2016. URL <https://nodejs.org/api/crypto.html>.
- [28] Node.js v6.0.0 documentation - timers. 2016. URL <https://nodejs.org/api/timers.html>.
- [29] Confidentiality, Integrity, and Availability (CIA triad). *WhatIs*, January 2015. URL <http://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA>.
- [30] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06*, pages 79–88, New York, NY, USA, 2006. ACM. ISBN 1-59593-518-5. doi: 10.1145/1180405.1180417. URL <http://doi.acm.org/10.1145/1180405.1180417>.