KTH Computer Science
and Communication

Anonymous Javascript Cryptography and Cover Traffic in Whistleblowing Applications

JOAKIM HJALMARSSON UDDHOLM

Master's Thesis at NADA
Supervisor: Sonja Buchegger, Daniel Bosk
Examiner: Johan Håstad

Abstract

In recent years, whistleblowing has lead to big headlines around the world. This thesis looks at whistleblower systems, which are systems specifically created for whistleblowers to submit tips anonymously. The problem is how to engineer such a system as to maximize the anonymity for the whistleblower whilst at the same time remain usable.

The thesis evaluates existing implementations for the whistleblowing problem. Eleven Swedish newspapers are evaluated for potential threats against their whistleblowing service.

I suggest a new system that tries to improve on existing systems. New features includes the introduction of JavaScript cryptography to lessen the reliance of trust for a hosted server. Use of anonymous encryption and cover traffic to partially anonymize the recipient, size and timing metadata on submissions sent by the whistleblowers. I explore the implementations of these features and the viability to address threats against JavaScript integrity by use of cover traffic.

The results show that JavaScript encrypted submissions are viable. The tamper detection system can provide some integrity for the JavaScript client. Cover traffic for the initial submissions to the journalists was also shown to be feasible. However, cover traffic for replies sent back-and-forth between whistleblower and journalist consumed too much data transfer and was too slow to be useful.

Contents

Chapter 1

Introduction

A whistleblower is a source who, usually secretly, exposes information about dishonest or illegal activity within an organization to the public eye. In a lot of countries whistleblowers are protected by law and journalists are required to protect the whistleblower's identity from inquiring parties. Hence the whistleblowing problem is the problem of how a whistleblower can anonymously communicate information to an agent of the public eye. Use cases range from news agencies taking in tips, to organisations looking to uncover internal corruption.

A famous whistleblower today is Edward Snowden [1, 2], who leaked information about US National Security Agency spying on large parts of Internet traffic [3]. Since then the whistleblowing problem has become a hot topic again. Although there have also previously been many cases throughout history where people came forward and exposed corruption anonymously.

Whistleblowers can run a large risk when they come forward. In many cases a compromise to their identity could change the rest of their lives for the worse. As such we are interested in protecting the whistleblower with anonymity. To facilitate an easier process for whistleblowing, we can attempt to engineer a system for it. In such a system we would then focus on primarily anonymity and ease-of-use. Whistleblowers should be able to leave tips anonymously and it should be easy enough to use that non-technical people feel comfortable using it.

Anonymity systems is a relatively new but studied area within cryptography. Applications range between classical problems like messaging and voting systems [4], to Internet routing (Tor [5], i2p [6]) and mixing virtual currencies [7]. This thesis evaluates current existing platforms which try to solve the whistleblowing problem and explores new features which improve the anonymity and integrity of the system.

1.1 Related Work

There are currently two open source project aimed at being whistleblowing platforms: GlobaLeaks [8] and SecureDrop [9]. Both systems work similarly by providing a web interface for both whistleblowers and journalists to connect to and interact via. They

also work primarily over Tor [5] and rely on the anonymity provided by it.

The technical details of GlobaLeaks, SecureDrop and also Wikileaks [10] will be examined later in the chapter on current implementations (chapter 5).

1.2 Thesis' Contributions

The thesis contributes to research by first looking at how current whistleblowing system implementations work. The systems looked at are the systems run by some Swedish newspapers, as well as SecureDrop, GlobaLeaks and Wikileaks. The systems are analysed based on privacy threats. The newspapers' systems are shown to suffer from several threats.

The thesis looks at how cryptography can be done in the client as opposed to on a server backend in order to improve confidentiality. The applicability of dummy requests is explored for their effectiveness to address issues with javascript integrity which arises from the use of clientside cryptography. Dummy requests are also explored for their use in providing timing and size metadata protection. These features are explored in the design of a new system, which is then evaluated based on the performance of smaller, partial prototypes made to test specific parts of the system.

1.3 Thesis Structure

The thesis is structured as follows. First, there is a prerequisites chapter with a background with some history around the legal situation in Sweden and some formal terminology. After that there is a chapter on the basic security requirements for a whistleblowing system. Then a Threat Model is constructed to set up threat actors and different privacy attacks which could affect such a system. With those attacks in mind, I look at and evaluate the technological features of current whistleblower platforms available by 11 different newspapers in Sweden. I also look at three existing platforms, SecureDrop, GlobaLeaks and Wikileaks, to briefly describe how those work and if they address the attacks earlier mentioned.

After that, I look at some suggestions on how to improve the current systems employed by the Swedish news agencies and some that may also apply to SecureDrop, Globaleaks and Wikileaks. With these suggestions I describe the theoretical design of a new system in the New Design chapter, which includes the use of javascript cryptography and dummy cover traffic. In the Prototype Implementation chapter I describe how different parts of this system were implemented and using performance results I estimate the supported number of whistleblowers, storage and transfer requirements for the system. In the Evaluation chapter I go back to the requirements and threat model established earlier in the report and evaluate the system. Finally in the Discussion and Conclusion, I discuss and state the outcome of the thesis; describing problems had, potential sources of error, other interesting approaches that could be taken and what the meaning of the results are.

Chapter 2

Prerequisites

This sections describes some prerequisite definitions and systems that the thesis will refer to later.

2.1 Whistleblowing in Sweden

This thesis looks specifically at the whistleblowing problem in Sweden. As such it is interesting to look at which laws are enacted which may support or hamper whistleblowing activities. This gives us a general idea of what a state actor is capable of doing in our current day.

In Sweden whistleblowers are protected by law in form of Source Protection [11] as part of Freedom of Speech laws. The law protects whistleblowers so long as the whistleblower themselves are not involved in very serious crime. Despite the protection offered by the Source Protection law, there has been at least one case documented where the Swedish Security Service (SÄPO) opened an investigation after a whistleblower leaked details about an FRA (the National Defence Radio Establishment) operation to the press [12].

Put into law in 2012, Datalagringsdirektivet (the data retention directive) says that service providers must save metadata about electronic communications. This metadata includes IP addresses, dates when the user was connected, from- and to-email addresses, Mobile identification in the form of IMEI and IMSI numbers, as well as personal identification numbers of the subscriber [13]. This logged data must be retained for at least six months. The purpose of the directive is to combat crime, allowing police to use this metadata for investigation in what are deemed to be serious cases of crime.

For Internet traffic leaving Sweden, the "FRA-law" allows the FRA to filter and analyze all non-Swedish communications [14]. The purpose of this law is to prevent terrorism and other threats against Sweden. Documents leaked by Snowden suggest that the security agencies of different states get around limitations set in law by trading traffic with other allied states [15].

Based on laws in Sweden, we see that the Swedish state itself, or at least agencies of it, can be seen as an adversary that has at least passive monitoring capabilities. It has

the capability to deanonymize IP addresses, and see which email addresses and phone numbers are communicating with each other. It also has the capability to passively eavesdrop on traffic passing Swedish borders.

2.2 Anonymity

Anonymity is defined by Pfitzmann and Hansen [16] as the state of not being identifiable within a set of subjects, the anonymity set. Unlinkability is the state of a subject not being linkable to an item-of-interest within a system [16]. An item-of-interest typically being a message sent between two users [16]. Unobservability is the state of a subject not being observable when interacting with the system. That is, we can not tell when one specific user is interacting with the system as opposed to any other user [16].

We can measure anonymity by the size of the anonymity set [17], which is the number of users that for an observer are identical in behaviour or interactions with the system. As such, when building an anonymity system we are interested in maximizing that number. For a website based service, our upper bound of such a set is the number of users on the Internet, because we know that the user had to connect to the service via that network. However, this can quickly become reduced by looking at other factors. For example when looking at a Swedish whistleblowing service, we can likely conclude that most users will also be from Sweden and therefore reduce the number of people in our anonymity set from the billions of the world to the 9-10 million people currently living in Sweden.

Another measurement of anonymity is the *degree of anonymity* defined by Diaz et al [18]. The *degree of anonymity* is a normalised value independent from the number of users based on the entropy of the system. It is calculated as follows.

$$d = \frac{H(X)}{H_M}, \tag{2.1}$$

Where H_M is the maximum entropy given N users and $H(X)$ is the entropy after an attacker has eavesdropped on communications. p_i is the individual probability that user i sent the message.

$$H(X) = -\sum_{i=0}^{N} p_i \log_2 p_i \tag{2.2}$$

$$H_M = \log_2 N \tag{2.3}$$

If we can determine the identity of the sender based on the contents of the message, then anonymity is dependent on confidentiality. Confidentiality can be defined as the property that information is not made available to individuals who are not authorized to see it. Anything which is supposed to be secret to outsiders should have the property of confidentiality.

2.3 Key-Private Encryption

Key-private encryption is encryption where the ciphertext produced can not be bound to the recipient [19]. As such an adversary investigating a ciphertext can not tell who it was for by looking at the encrypted packet alone. This excludes PGP as an key-private encryption method, since it includes the intended recipient's public key fingerprint in the packet data [20]. Also RSA is excluded, since the ciphertext bytes produced would likely be biased towards different distributions depending on the n modulo used, which usually differs depending on the keypair used to encrypt.

Key-private encryption may also require key robustness [21]. Robustness requires that it is difficult to produce a single ciphertext that is valid for two different users. If a ciphertext can be decrypted into different plaintext messages for different users, it may be difficult to tell which user the message was meant for, and which message content was correct.

2.4 The Tor Anonymity Network

The Tor anonymity network [5, 22] is a distributed, anonymous network operated by volunteers that allows users to connect to the Internet anonymously.

A *circuit* is a random path of nodes that a Tor client chooses for it's user to route it's traffic through. The circuit consists of at least one *entry-relay*, one or more *middle relays*, and at least one *exit-relay*. The entry-relay is where the packets first enter the network from the Internet. The middle-relay simply forwards packets to their next destination in the network. The exit-relay is where the packets exit the circuit and will be the IP address that is presented to the final destination.

Each packet is wrapped in layers of encryption. One layer for each node that it passes through. Each node unwraps one layer of encryption and sends it to the next destination specified in the packet. The system works in such a way that only the entry node knows the origin of the packet, but it does not know where the packet is headed. The exit relay can see the destination of the packet and sometimes the contents, but can not see where the message came from. Only if relays collude can they distinguish both the packet source and the destination. The security of the Tor network relies on this fact that with many relays it is difficult for an adversary to control of both entry and exit relays of a given circuit.

In addition to routing connections to IP addresses on the Internet, Tor also features *hidden services*. These are designed to be access by the use of a specific onion-domain and are meant to be used for anonymous servers. Onion domains are determined by the fingerprint of a cryptographic certificate, meaning the authenticity of the server can be proven by the fingerprint. A client connecting to an onion url can verify the integrity of the connection to the service with the received certificate by comparing this fingerprint. When setting up a hidden services, the server administrator chooses a set of guard nodes for that service, which will be used to forward the connection to the server. Hidden services are often found used for illict purposes, such as marketplaces for drugs,

but also for legal activities involving activism, and whistleblowing services.

The Tor browser [23] is a custom Firefox installation used for browsing the web that comes pre-configured with a standalone Tor instance which it routes all network traffic through. It also comes equipped with extra plugins, such as HTTPS Everywhere [24] and NoScript [25], to further increase privacy of users. The Tor Browser is specifically configured to protect the users privacy and identity.

2.5 Cover Traffic

Part of the thesis explores the usage of cover traffic which has been studied previously. Research by Mallesh-Wright [26] classifies cover traffic for mixnets (a type of network which randomly shuffles in- and outgoing packets) into 3 types based on a model of *senders* and *receivers*. *User cover* is traffic generated by the sender, *background-cover* is generated by other senders and *receiver-bound cover* is generated by the mix and sent to receivers. Mallesh et. al. show that *receiver-bound cover* is effective at preventing or at least delaying statistical analysis.

While this thesis' proposed system does not include a mixnet by itself, it does use Tor which has similar properties to a mixnet. In the sense of differently bound cover, we will later see that the system in this thesis uses receiver-bound cover, because the cover traffic is generated by the system's receivers.

2.6 Acronyms

FRA National Defence Radio Establishment

IMEI International Mobile Station Equipment Identity

IMSI International Mobile Subscriber Identity

TOTP Time-based One-time Password

IETF Internet Engineering Task Force

DNS Domain Name System

GPS Global Positioning System

Chapter 3

Threat Model

This chapter attempts to summarize the threat actors and the threats that they possess against a whistleblowing system. After that there is a section on specific threats that apply to a whistleblowing system.

3.1 Threat Actors

In a perfect world, an anonymous messaging system would be simple to model. A direct channel between the whistleblower and the journalist, that is both secure and anonymous would satisfy the requirements. In such a model, only two people would ever interact with each other.

However, in practice true peer-to-peer functionality is hard to achieve. For example, when using the Internet, there is also the numerous hosts in between the users forwarding the packets. At higher level protocols, such as bittorrent or even Tor, there is also either a directory or tracker that the user must connect to first in order to determine where the other users or relays are.

In addition, computers operated by actual humans and that are not servers have a tendency to be offline and sometimes move physical locations. For usability and availability reasons, the system is often built with at least a third party running as a server. Figure 3.2 shows the very basics of such a model.

This thesis focuses on Figure 3.3, which shows the overview of the threat model, with the actors and network zones. For each link between the parts of the system,

Figure 3.1. The wanted model for a whisteblowing system

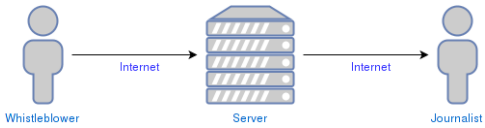

Figure 3.2. A more practical model for a whistleblowing

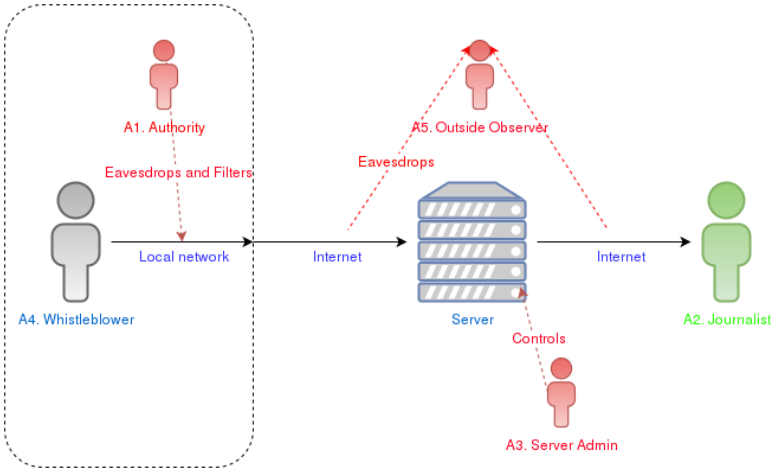

Figure 3.3. The threat model

represented by arrows, we can assume there are a number of hosts. Based on this model, we can already see that we have five different threat agents: *Authority*, *Journalists*, *Server Admins*, *Whistleblowers* and *Outside Observers*. All of these agents have the possibilities of acting maliciously.

With the nature of a leak we assume that there is some adversarial actor that is against the data of the leak being made public, and that there could be some consequence if the identity of the whistleblower is disclosed to this actor. It could be that an employee leaking data from an organisation risks getting fired, or a soldier leaking information about a state army risks imprisonment for espionage. To refer to this adversarial actor I will simply call them the *Authority*. This authority could be in control of the network that the whistleblower connects from. In this case, they have the capability of passively monitoring outgoing traffic for exfiltration or in some cases even actively trying to man-in-the-middle traffic that goes in and out of the network. However these capabilties only extend as far as the network that they control. It is assumed that the authority does not have these capabilities on the same network that the server runs on. The whistleblower may also be submitting from a different network (e.g. not at work), in which case the Authority has no adversarial capabilities against the whistleblower.

A malicious journalist can attempt to disclose the identity of the whistleblower. This was seen for example when the journalist and former hacker Adrian Lamo disclosed the

identity of whistleblower Chelsea Manning to US authorities, resulting in the imprisonment of Manning [27]. This could also be done unintentionally by negligence, such as if a journalist does not handle the data correctly. As an example of such negligence, the antivirus millionare McAfee was apprehened by police after a picture taken by journalists leaked GPS coordinates [28] in the metadata. Thus we can not assume that journalists will never break confidentiality. However, for the purpose of this thesis protecting the identity from being disclosed by the journalists is out of scope, since it likely can not be solved by an engineering approach, but rather by organisational security, policies or law. As mentioned earlier, the Source Protection law is one way of addressing this problem. As a journalist in Sweden, you risk a jail sentence if you disclose a whistleblower's identity.

For this thesis and the system I construct, I will work with the assumption that journalists are *trustworthy* and do not wish to deanonymize whistleblowers. As such, threats that come from a journalist attacker are not considered.

Server administrators make sure that the server that runs the whistleblowing application keeps running. As such they will always have the ability to remove that availability. In addition, they may be able to monitor the usage of the service. The server admins can see what IP addresses connect to the service, what time they connect, and what data is submitted. They can also modify the behaviour of the server or leak information to a different actor. By the nature of any system that relies on a third party to host a service, that third party always has the technical capability to deny that service.

I will make the assumption that server admins can be trusted enough to keep the service running, but not that they will not try to tamper with it or leak information from it. They could, for example, under a secret court order (see National Security Letters [29]) have to allow secret access to the server to secret agents. For this reason, the closer we get to an *untrusted server model*, the better.

The whistleblower themselves might try to interact with the system in such a way to disclose the identity of other whistleblowers or gain access to leaks submitted by other whistleblowers. They could submit malicious files that attempt to exploit the server or journalist machines to elevate privilege. Because of the anonymity requirement for whistleblowers, any one of the other threat actors could also be whistleblowers. As such they have the same capabilities of that of a malicious whistleblower. Extra precautions should be taken against attacks that can be mounted by malicious whistleblowers. It must be assumed that whistleblowers can also be malicious, and will try to compromise the interface given to them. As such, whistleblowers are not trusted to always behave correctly.

Outside observers are those that do not necessarily interact with the system. These threat agents could be hosts that route the Internet connection between parts of the system, hosts on the same local network, or nation states. For the purpose of this thesis, we look at the Swedish state as a passive monitoring adversary with capabilities to look at retained data. In the event of a breach against national security, they may have the capability of seizing a server or, as mentioned before, looking at Internet traffic going back 6 months. While the Swedish state may not be interested in actively trying to

monitor usage of a specific whistleblowing service, they may have the capability and sometimes interest (in the name of national security) to go back in time and look at traffic to a specific service retroactively. Assuming the worst, we have to assume that some outside observers are at least able to passively surveil parts of the Internet. Outside observers are assumed to be *honest-but-curious* as they will likely not interact directly with the system, but they will bulk collect data that passes through it and later try to unmask whistleblowers using that data. The state actor is included in the classification of outside observers.

3.2 Privacy Threats

This section describes a number of ways privacy can leak in an Internet based whistleblowing platform. Most of the leaks described in this section are focused on a website based application, which seems to be the most common type of application used to facilitate initial communication between whistleblowers and journalists. These threats will be used to identify vulnerabilities in the chapter on Current Implementations (chapter 5) and to evaluate the new system in the Evaluation chapter (chapter 9).

3.2.1 IP Leak

Connecting via normally routed TCP-IP means the client has to provide an IP address for the server to reply to. Unless the client is on a different network, either virtual or physical, this IP address is likely linkable to their real identity. Referring back to the earlier described Data Retention Directive that exists in Sweden, it is possible that state police might be able to get records of who connected to a specific IP address (e.g. the whistleblower website) from the hosting internet service provider. As such, for a whistleblower concerned about a state actor, precaution has to be taken to not use an IP address that can be bound to the whistleblower's real identity.

3.2.2 DNS Leak

If the platform is hosted on a separate domain name, this introduces a new problem against adversaries eavesdropping against DNS requests. Since DNS requests are not encrypted, it is easy for an adversary who can eavesdrop on a network to determine which IP addresses are connecting to the whistleblowing website. If the domain is hosted exclusively for whistleblowing purposes and does not attract visitors for other reasons it increases the likelihood that a user performing a DNS lookup for the service is indeed a whistleblower.

As an example, take the domain "securedrop.newspaper.com". Any visitor to that website could be marked as a potential whistleblower for just interacting with that system. An external eavesdropper can identify a more narrow set of users who are more likely to be whistleblowers than say those who would visit "www.newspaper.com".

Figure 3.4 shows the example in question. The full DNS request is seen by at least two different DNS servers, as well as any network eavesdroppers on those links. In addition,

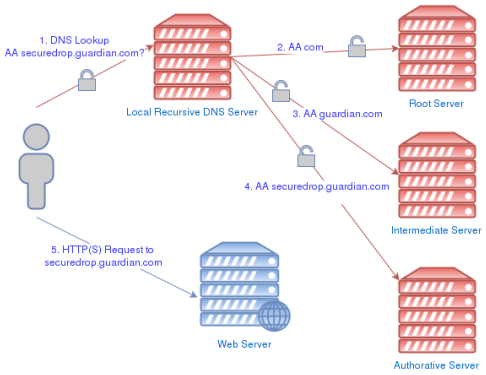

Figure 3.4. Typical DNS lookup when connecting to web server

it would be easy for a network adversary (the Authority) to perform a man-in-the-middle attack if the local recursive DNS server does not use some form of authentication, such as DNSSEC.

3.2.3 URL Leak

Related to the DNS Leak, if instructions or the whistleblowing service itself are put on a specific page URL, e.g. "www.newspaper.com/how-to-whistleblow", the set of visitors that accessed that URL might be correlated to the people who submitted stuff to the whistleblowing service. As such a server administrator might be able to see the real IP addresses, which could in turn reveal the identity of the whistleblower. If the connection is made over an unencrypted channel any network eavesdroppers also see the URL.

3.2.4 Lookup Leak

A system using anonymous channels might on the surface be anonymous, but misses the point when whistleblowers initially seek information on how they should communicate with the system. The steps taken before interacting with the whistleblowing system are important too. Typically they have to download a specific client, or follow a set of instructions given by the newspaper. Doing this itself on a not-anonymous channel may implicate them by just accessing that information. The DNS Leak, IP Leak and URL leak are all versions of this leak. A user that accessed instructions on how to whistleblow can be correlated by time to the submission of a leak.

3.2.5 Software Usage Anonymity Leak

Systems that rely on clients using specialized software that has to be download and installed separately provide a threat to anonymity too. Firstly, the same Lookup leak applies from before; users that install a specific software have to get that software from somewhere. In doing so, they may end up on a list. As an example, there is spec-

ulation around whether the NSA tracks Tor and Tails installations [30]. Second, the communications protocol used by the software might be uncommon or not anonymous by itself, providing less anonymity than say HTTP would simply because the usage of that protocol being rare.

3.2.6 HTTP Confidentiality and Integrity

The HTTP connection to the webservice must be done over HTTPS or there will be no confidentiality or integrity and therefore no guarantee for anonymity. Without using TLS for HTTP, clients are open to eavesdropping or man-in-the-middle attacks. A malicious network attacker could for example inject extra HTML markup into the content of the website to serve malicious javascript code that calls home to the adversary, or it could directly serve exploits that attempt to bypass the browser into the running operating system.

There is also another risk in mixing encrypted content with unencrypted content. If the server provider decides to serve the only whistleblowing service over HTTPS and the rest of the site over normal unencrypted HTTP, then the anonymity set is limited further to only those that connected to the site using the HTTPS protocol on an IP basis. Without HTTPS the contents are sent in cleartext and there is no confidentiality against network observers, such as the Outside Observers and the Authority. This also assumes that when using HTTPS the TLS cryptographic certificate is properly signed and validated against, as otherwise it is very easy to perform a man-in-the-middle attack.

3.2.7 Third Party Services

Scripts hosted on external domains, such as popular javascript frameworks, advertisement networks or user tracking, increase the risk of a privacy leak. Most of these external services are advertisement networks and other forms of tracking generally used to perform website analytics. Assuming a large provider like Google can track which URLs are visited by a user visiting the website, and that the same provider also has other metadata stored which identifies the target user. Then it becomes clear that such providers could have information about which users might be potential whistleblowers based on the URLs visited. They will at the very least contain information about which other websites were visited by the user, which can give away the user's interests, location and other parts of their identity, thereby removing anonymity from whistleblowers.

Loading third party content from the HTML will by default include a Referrer-header in the HTTP requests pointing to the URL which was visited. This is why many tracking companies simply work by using a small empty image on the target website, because they can track using the Referrer-header alone.

Search engines are a particularly difficult form of tracking to avoid, as any user that would look up the whistleblowing service in a search engine would likely also be tracked when clicking the link to that website. For example, by default Google saves the history of user searches which means they would have information on who clicked links going to whistleblower websites.

In addition to privacy leaking via tracking analytics, any external hosted script runs the risk of running malicious javascript which in turn could track when any whistleblowing action is performed. For example, a malicious script submit a copy of any data sent to the newspaper to an eavesdropper. Even if the script in question is benign it could be transferred over unencrypted HTTP. In this case a network adversary could again perform a man-in-the-middle attack to inject their malicious javascript, as described under *HTTP Confidentiality and Integrity*. The more third party services that are used on a domain, the more trust must be had that they are all benevolent as well as secure.

Even if the newspaper does not use third party services on their whistleblowing page, it is likely that the lack of tracking itself can be a be used to fingerprint which users over a network fetched a whistleblowing contact page. If the website normally features a lot of tracking, a request that does not follow other requests to tracking websites will stand out to a network observer.

3.2.8 Non-anonymous Emails

While it is possible to send emails anonymously, it requires a lot of effort from the user. Achieving email confidentiality might be difficult to begin with, but it is possible using PGP [31] (Pretty Good Privacy) or other types of encryption.

Unless encrypted, contents of emails are sent and stored in plaintext. In addition, email has a from-address as an obvious indicator of who sent the email, as well as headers which can provide IP address of the user who sent the email. By default, email is therefore not anonymous nor confidential. However, it is possible to attain anonymity by using anonymous email services. Either by simple temporary hosted email solutions, or more complex mixnet based remailers.

Different capable adversaries could eavesdrop. The Authority adversary in control of the local network could be logging all outgoing emails. The email provider is most likely by protocol definitions saving all sent and received emails in plaintext. The network for sending the email from the whistleblower's computer through to the journalists' computers has multiple hops. Any of those hops could eavesdrop on the traffic if it is not sent over an encrypted channel. In addition, even if the email is encrypted, most implementations for email encryption including PGP leave the header fields and subject line unencrypted.

3.2.9 Server Confidentiality and Integrity Threats

Data that is stored or transmitted unencrypted at any points in the system may be read by agents that are not supposed to access that data. In a whistleblowing system involving a server, a malicious server administrator may have access to confidential data if it is not encrypted prior to being sent to the server. At this point we can only trust that the server administrator does not read or copy that information themself, or that the server does not get compromised by an adversary in the future.

The threat can be split into two. The first threat regards data retention, and regards data which might be stored for a longer period of time unencrypted. In this case an ad-

versary which compromises the server could gain access to past leaks. The second threat applies to data that is encrypted on submission. In this case the server administrator, or a more patient adversary, could subvert the data prior to encryption.

3.2.10 Traffic Analysis Leaks

Assuming that the service is monitored by an adversary, the adversary may have access to information about timing, for example when a leak is submitted, or the size of a leak which can be used to deduce certain things about the leak. An organisation that is monitoring internal traffic and traffic going to or from the whistleblowing service could correlate these factors of submissions even if they are encrypted.

3.2.11 Who-Had-Access Leaks

Other analysis in regards to who might have had access to the leaked material or timing might also reduce the anonymity set significantly. For example if a document is classified in a military setting so that only a select few can see it, then likely only one of those select few could be the ones to leak that document.

Different tactics can also be used to detect the source of data exfiltration, such as embedding fingerprints into the document files depending on who sees the original document. Using steganography this process can become very difficult to detect, especially if you only have one version of the file.

Chapter 4

Requirements of a Whistleblowing system

The minimum goal of a whistleblowing system is to be a system where whistleblowers can send data anonymously to one or more journalists. The main focus of the system is to protect the identity of the whistleblower. This section describes different security requirements that are important to a whistleblowing system.

4.1 Anonymity for whistleblowers

Whistleblowers are assumed to be at risk when divulging information. Therefore steps have to be taken to protect their identity. As such it is a requirement that the whistleblowing system takes steps to ensure anonymity for whistleblowers.

A whistleblower system would preferably be sender unobservable [16]. In such a system, senders have the unobservability property; they can not be seen interacting with the system. If a user is identifiable as being part of the set of whistleblowers that can also be enough to jeopardize the users safety. That is, if only whistleblowers are the users of a system, then usage of that system by an observed user is a strong implication that the user is a whistleblower. If you design a software where the anonymity set for users is only a certain type of user then it is obvious to correlate a person's usage of that software into them being that type of user. Preferably, the set of users should include other types of users too in order to give plausible deniability to any observed user.

4.2 Integrity

Integrity is the property that ensures that the process or data that is transferred in the system can not be changed by someone unauthorized. This means that the application should not be easy to compromise through bad security, but also terms like trustworthiness come into play. The process must be transparent enough that users can trust the system.

Anonymity is dependent on the security of the application. An insecure application might host attacks that compromise the anonymity, availability or confidentiality of the system. As an example, if the server that hosts the system is compromised an attacker

could add malware which tries to compromise client machines, which in turn could reveal the identity of client users. Another attack could be against the confidentiality of the application by removing any encryption used. Part of integrity is the assurance that systems do what they claim to do, even when under attack by an adversary.

4.3 Confidentiality

Confidentiality is a property similar to privacy. Information that is confidential should not be accessible to those without authorization. Information shared between whistleblower and journalist must be confidential as it could be sensitive when in view of outside eyes. The data shared by the whistleblower could withhold the identity of the whistleblower themselves or contain information that could put people in danger. Messages sent between whistleblower and journalist must be kept secret, both in terms of contents and metadata around those messages. The contents of the messages reveal anything the whistleblower said about themselves, and metadata can place the whistleblower at a certain time at a certain place. Messages should not be linkable back to the identity of the whistleblower. If the whistleblower chooses to reveal their identity to the journalists, that piece of information should still be confidential, at least within the confines of the whistleblowing system.

4.4 Availability

Availability is the property that ensures that the system is available. This might mean that in practice, it does not go down due to system errors or even purposeful denial-of-service attacks.

The information submitted to the system is both critical and sensitive. It could for example detail a corrupt politician leading up to an election, or a corporate coverup of an oil spill. Such information can be important to get out as quickly as possible to the public. Therefore it is required that the application is available as much as possible, or there is a risk such information is lost.

4.5 Usability

Usability entails how easy and intuitive the application is for a user to use. Ensuring that the process is not painful promotes users to use the system rather than less secure or anonymous means. In addition, by the nature of anonymity: the more users, the better anonymity. The system must be usable for both whistleblowers and journalists. Whistleblowers must be able to use the system without the protection features getting too much in the way, and journalists must be able to use the system without it interfering too much with their work routines. I will make the claim that regardless of how secure the system may be, it must also be usable for it to have any value.

Chapter 5

Current Implementations

This chapter looks at current implementations of whistleblowing systems. The first section summarizes the offering from newspapers and other media organisations in Sweden. The second section looks at the existing open-source implementations SecureDrop and GlobaLeaks, and the whistleblowing service Wikileaks.

5.1 Survey of Swedish Newspapers Whistleblowing Platforms

Eleven different newspapers in Sweden were evaluated based on their capability to receive anonymous tips over the Internet. These were picked from the Alexa list of the top domains by number of visitors in Sweden [32]. Table 5.1 shows the newspapers evaluated and where the interface for submitting tips could be found.

Of the eleven newspapers, *Expressen*, *Svenska Dagbladet*, *Dagens Industri* and *Resume* did not have any website as a service for whistleblowers. Instead they list either email, phone calls or SMS that the whistleblowers can submit tips through.

Table 5.2 looks at the seven newspapers that did have a web based service for whistleblowing and whether or not certain privacy features were implemented. In the separate domain column, having a separate domain means being vulnuberable to the DNS leak. The clientside encryption column checks if the contents of the leak is encrypted before submission. The PGP key column is checks if there was a PGP key available for use for whistleblowers who wish to manually encrypting leaks. The external domains column shows the number of separate outside domains that were included upon loading the whistleblowing service, exposing the service to the threat of a third party leak. The Tor hidden service column shows whether or not the service was available via a hidden service, which could help in avoiding the IP leak.

Of the seven newspapers who did have some sort of website for sending in tips, *Sydsvenskan* and *Metro* served their website over an unencrypted channel without HTTPS. If this traffic is logged, it could be easy for the Authority or the Outside Observer adversaries to deanonymize whistleblower traffic since it is sent in plaintext.

Five of the newspapers run their whistleblowing platform on a separate domain from the normal newspaper website. This means they are susceptible to the DNS leaking

weakness described earlier.

Four of the newspapers, *Aftonbladet*, *Metro*, *Sydsvenskan* and *Göteborgs-Posten* include a lot of third party content hosted on external domains on their submission site. This means that these third party providers could potentially track which users are making submissions or visiting the whistleblowing website. This could potentially deanonymize the whistleblowers to these services.

Dagens Nyheter and *Sveriges Television* make use of a PGP-implementation in Javascript to encrypt contact details before they are sent to the server. However, they do not go all the way and leave the file contents unencrypted before submission to the backend server.

The study also tried to find Tor hidden service connected to the newspapers' whistleblowing services. However, none of the newspapers supplied any onion-addresses so it was concluded that none of them use a Tor hidden service.

After further investigation, it was found out that the Radioleaks, TVLeaks, and Techleaks share the same provider company Bahnhof, which is likely why they have similar implementations. Other newspapers were not forthcoming in regards to details about their implementations, and some chose to not discuss due to security reasons.

Table 5.1. Newspapers Studied

Alexa Rank (Sweden)	Media Outlet	Domain	Whisteblower URL
5	Aftonbladet	`www.aftonbladet.se`	`https://www.aftonbladet.se/tipsa/`
11	Expressen	`expressen.se`	None
17	Dagens Nyheter	`dn.se`	`https://dngranskar.dn.se/`
19	Svenska Dagbladet	`svd.se`	None
26	Sveriges Television	`svt.se`	`https://tv-leaks.se/`
30	Dagens Industri	`di.se`	None
51	Sveriges Radio	`sr.se`	`https://upload.radioleaks.se`
68	Göteborgs-Posten	`gp.se`	`https://www.gp.se/nyheter/tipsagp`
110	Sydsvenskan	`sydsvenskan.se`	`http://apps.sydsvenskan.se/tips/`
135	Resume	`resume.se`	None
138	Metro	`metro.se`	`http://www.metro.se/om-metro/tipsa-metro/EVHnis!tXEUOQqWxxpcA/`
445	Nyteknik	`nyteknik.se`	`https://techleaks.se`

Table 5.2. Privacy Features

Media Outlet	Separate Domain	Clientside Encryption	PGP key	External domains	Tor Hidden service
Aftonbladet	No	None	None	8	No
Dagens Nyheter	Yes	Partially, file content not encrypted.	Yes*	0	No
Sveriges Television	Yes	Partially, file content not encrypted.	Yes*	0	No
Sveriges Radio	Yes	None	None	0	No
Göteborgs-Posten	No	None	None	18	No
Sydsvenskan	Yes	None	None	14	No
Metro	No	None	None	13	No
Nyteknik	Yes	None	None	0	No

5.2 Existing Frameworks

Today there are two existing open source whistleblowing applications: the GlobaLeaks [8] and the SecureDrop [9] projects. Newspapers who wish to run a serious whistleblowing service have a good ground to begin with if they host one of these applications. Most of the problems described earlier in this thesis are solved by running either of these applications. This section will go more into detail of these two solutions as well as WikiLeaks [10], which is an organisation that specializes in handling leaks.

5.2.1 SecureDrop

SecureDrop is an open-source whistleblower submission system that media organizations can use to securely accept documents from and communicate with anonymous sources [9]. SecureDrop is developed by the Freedom of the Press Foundation and used by some of the biggest newspaper organizations in the world including The Guardian [33] and The Washington Post [34].

The SecureDrop application itself is a web application coded in Python, but it is built to be part of a larger system where the application is just part of it. The application comes pre-packaged with a variety of security applications preconfigured to be easily set up on its own server. Figure 5.1 shows the overview of the system architecture recommended by the SecureDrop team.

SecureDrop uses two Tor hidden services. A public service for the sources to use and a private authenticated service for the journalists to use.

PGP is used to encrypt the leaks, however only after they have been received by the server. No clientside encryption is done, although the website recommends the whistleblower to encrypt the leak with the journalist key manually before submitting. This functionality is not provided by SecureDrop itself.

The interface itself is quite simple. As a whistleblower, the user has access to a form to upload files and to leave a message. The whistleblower is also assigned a 7-10 word long codename as a pseudonym which can be used to authenticate the whisteblower if he or she decides to come back to the interface later. This feature is mainly there to support a conversation between the whistleblower and journalist.

The journalist interface requires password and two-factor authentication to log in. Once logged in, the journalist is presented with a list of sources and their submissions. Each submission shows a thread of downloadable messages and files encrypted with GPG for the journalist and has a form to send back messages to the whistleblower. Any messages must be downloaded and decrypted by the journalist manually. SecureDrop does not provide the decryption functionality for the journalist.

SecureDrop is designed to work with Tails [36], an operating system built around Tor and for the purpose of keeping its users anonymous. SecureDrop recommends both whistleblowers and journalists to use this operating system, preferably on a live-USB. Running the operating system on a live USB is meant to prevent data retention and make usage less bound to a specific computer (i.e. portable).

SecureDrop puts a lot of emphasis on security. It features mandatory two-factor

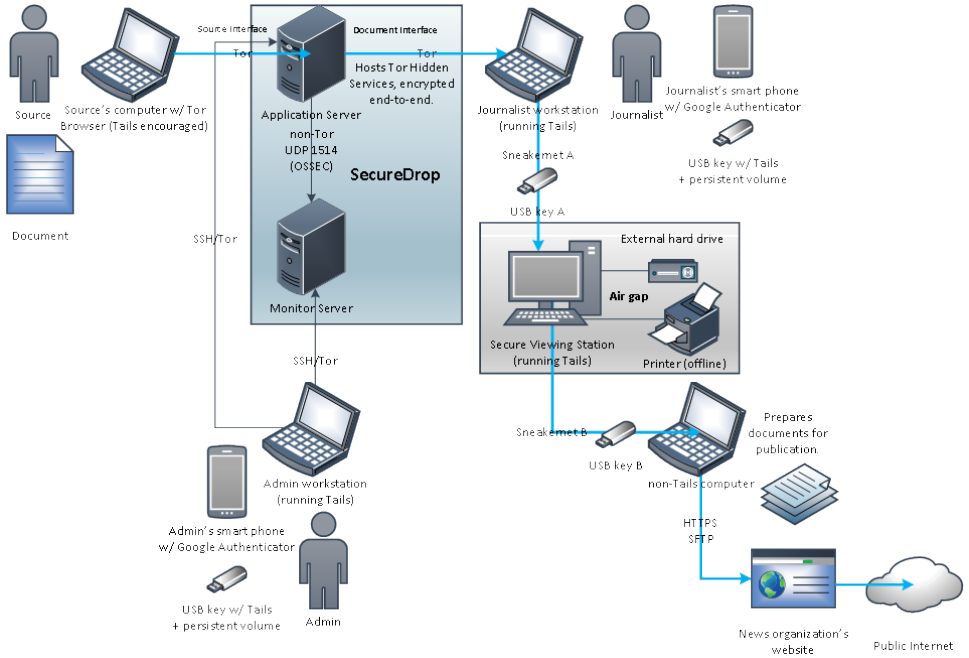

Figure 5.1. SecureDrop architecture (taken from [35])

authentication for the journalists in the form of TOTP (using Google Authenticator). It uses several server side frameworks to harden security of the application itself. This includes GRSecurity [37], OSSEC [38], iptables and AppArmor [39]. GRSecurity is a specialized linux kernel patched for security. OSSEC is an intrusion detection system. IP-tables is a commonly used firewall for Linux. AppArmor is a kernel security module that allows for per-program placed restrictions. All of these security features are interesting by themselves, but will not be covered extensively here. The strong security featured by SecureDrop helps prevent against attacks which could break anonymity.

Furthermore, a lot of emphasis in SecureDrop is put on preventing data retention. We see this with the aforementioned recommended usage of Tails, but also the feature of a secure viewing station. The secure viewing station is a purpose built computer for decrypting leaks sent to the SecureDrop application. The secure viewing station has no access to the Internet and no harddrive to save long term data on. Submissions have to be physically transferred with a USB stick or some other method to the viewing station. This creates an air-gap, which is a security feature designed to prevent advanced attackers who already gained access to the network from gaining access. This way the plaintext is hypothetically only ever seen on the viewing station. In addition to the viewing station, we also see the recommendation of a dedicated printer that is also offline. This way the journalist can take the PGP encrypted data to the viewing station, decrypt it using his/her live-USB stick and then print it out. This makes it almost impossible for

a network attacker with full eavesdropping capabilities to determine the contents of the leak. Because the secure viewing station is also built without a harddrive, that makes it even harder for an attacker to persist on the machine by the means of malware or for journalists to accidentally save the data unencrypted.

Strengths and Weaknesses

SecureDrop goes to great lengths in order to avoid non-encrypted content being leaked within its whistleblowing system. It uses Tor and has strong security features backing it.

From looking at the list of SecureDrop instances [40] it seems like many instances have a separate domain name host. This could lead to whistleblowers being fingerprinted by the DNS traffic as described earlier. The second issue is that the encryption is done serverside, which I will with this thesis argue could be done in the client. The third issue concerns the usability of the service for journalists. Since they are required to download each GPG message separately, decrypt and verify the contents manually. This process could take a lot of work and is prone to abuse by spammers.

5.2.2 GlobaLeaks

GlobaLeaks is an open source project aimed at creating a worldwide, anonymous, censorship-resistant, distributed whistleblowing platform [8]. Like SecureDrop, GlobaLeaks is a webbased platform made to run over Tor. The GlobaLeaks project is maintained by the Hermes Association. The same association also maintain projects related to Tor, such as Tor2web [41], a proxy for normal web traffic to reach Tor hidden services [42]. GlobaLeaks is a website framework written in Python and JavaScript.

GlobaLeaks displays a clear warning for the user when it accesses the site without using Tor, and forces the user to answer a question about the anonymity and integrity of the protocol used to connect. This way it puts effort in trying to protect the user by the way of education. This safeguards non-technical users who may not otherwise know the risks.

The interface for GlobaLeaks is somewhat more advanced than that of SecureDrop. It allows the whistleblower to pick which journalists he wishes to disclose to, and the backend then encrypts the contents using the chosen journalists' keys. This way a whistleblower has more control over who can read his data. Upon submission, the whistleblower is assigned a 16 digit code for use when returning to the whistleblowing service.

GlobaLeaks encrypts the contents of a leak serverside using a similar scheme as PGP. There is as of the current version no clientside cryptography in GlobaLeaks, but there is a feature in the making which means to add end-to-end functionality to the platform [43]. GlobaLeaks does try to prevent any types of plaintext data from being saved on the machine by temporarily encrypting the received content using AES, before a separate process encrypts it with PGP. This is done so that the file content never is saved to disk without being encrypted, and because the PGP encryption might take some time

to complete. If an adversary were to shut down the server before the PGP-encryption happens it may otherwise have been able to read the unencrypted file from disk.

By default GlobaLeaks will automatically set up a Tor hidden service and is not designed to work outside of Tor. Instead, users wishing to access a GlobaLeaks node outside of Tor are encouraged to use a Tor2Web proxy.

Strengths and Weaknesses

GlobaLeaks puts emphasis on using Tor to connect and differs from SecureDrop by allowing the whistleblower to handpick which specific journalists should receive a leak. It attempts to educate users on the privacy of HTTPS, as well as displays a big warning when not using Tor in order try and get the whistleblower to improve its own anonymity.

Proxying via Tor2Web is used as a solution for those that do not want or can not connect via Tor. However, use of this proxy is susceptible to DNS leaks and users will not be anonymous to the operators of the Tor2Web proxy.

5.2.3 Wikileaks

Wikileaks is a not-for-profit media organisation [10], famous for its published whistleblowing stories. Since Wikileaks is a service and not a framework, the source code for Wikileaks is not publicly available. However, the submission process can still be observed from the client side and is therefore still of interest for this report.

Wikileaks strongly enforces usage of Tor for its submission form. The form is only accessible via a Tor hidden service, meaning a whistleblower has to use Tor to use the submission form. Additionally it only displays the onion address via an anchored URL: `https://wikileaks.org/index.en.html#submit`, thus preventing URL and DNS leaks during lookup.

However, unlike other news agencies Wikileaks is perhaps not as commonly visited for reading news, but rather specifically leaks. Thus it might be likelier that the website is under scrutiny by organisations and states.

As for the actual form, like SecureDrop and GlobaLeaks the submission form does not support client side encryption but does claim to encrypt the form serverside. Wikileaks instead provides a GPG key for whistleblowers to use in manually encrypting leaks. Wikileaks also asks the whistleblower to answer a set of questions. These range from regarding who had access to the information, what the threat is to the sources of the leak and when the leaks should preferably be publicized. These questions likely help Wikileaks staff to determine the level of operational security necessary in handling the submitted content.

Strengths and Weaknesses

Wikileaks enforces Tor usage by whistleblower and by design hides metadata such as the submission-lookup URL.

The enforced usage of Tor forces users to download the Tor browser. The transparency of the site is lacking, since the encryption happens serverside. It does to some extent prevent the DNS leak issue by the use of an anchored URL. However, since it likely does not have the same factor of deniability that a normal newspaper has, visiting the main domain of *wikileaks.org* alone might fingerprint a potential whistleblower.

Chapter 6

Suggestions for Improving the Current Systems

This section describes some suggestions and general improvements that can be done for whistleblowing services. Based on the study performed in the last chapter, it could be used to improve current newspapers' whistleblowing platforms.

6.1 Prevent DNS and URL Leakage

In regards to the Data Retention Directive, Swedish newspapers should attempt to move away from whistleblowing solutions with separate domain names. DNS traffic is not confidential and so I would suggest that newspapers instead host on the same domain as the normal newspaper content. So instead of "securedrop.newspaper.com" it could be hosted as "www.newspaper.com/securedrop/". The latter URL would not trigger a different DNS query and therefore not leak potential metadata about whistleblowers into the domain name system.

Wikileaks avoids DNS leakage by enforcing usage of a Tor hidden service. When looking up where to find a form to upload/contact wikileaks on the main website, the link is displayed only in a JavaScript triggered popup and has no server interaction to open it. This also prevents leaks from specific URLs being visited, not even the server itself can see if a user looked up information on how to access the whistleblowing service.

A solution for a linkable URL could be to have the whistleblowing form or information about it available in an anchored URL, such as "www.newspaper.com/#securedrop". In this case the server hosting the newspaper would only see a normal request to "www.newspaper.com" with the "#securedrop" part omitted.

Another alternative could be to inject references to the whistleblowing domain into normal webpages on the content domain. This way non-whistleblowers just visiting the website would also send DNS queries to the whistleblowing site.

In the future we may have something like DNS-over-TLS to provide privacy for DNS, which is currently being developed by the IETF DPRIVE working group [44], but until then it should be assumed that DNS lookups can be eavesdropped on.

6.2 Tor Hidden Service

Using a Tor hidden service promotes users to use Tor, which is effective at providing anonymity. In addition, hosting a service exclusively over Tor makes it difficult for adversaries to seize the actual physical server where the whistleblowing service is kept. Whistleblowers may not be entirely aware of how identifiable they are when they submit tips over the Internet, so enforcing usage of Tor may help prevent whistleblowers from making mistakes which could compromise their anonymity to the server admin, the authority or the outside observers.

Most of the privacy issues described earlier in regards to IP, DNS and URL leaks are solved by using Tor. However, in order for Tor to work the user needs to be using it as early as possible in the whistleblowing process.

There are some caveats with using Tor, as mentioned earlier by the software usage leak it may be possible for the authority adversary to identify a potential whistleblower just by the fact that they are using Tor. However, the use of Tor is in my opinion a better case than not for privacy and anonymity use. The IP, DNS and URL leak attacks that it prevents are more severe than those that come from using Tor. In addition to this, there are bridges that can be used to proxy to Tor that mask the origin traffic coming from the Tor clients in something more innocuous such as HTTP.

6.3 Encrypted Leak Contents

Using an encryption scheme like PGP works to limit which people have access to the leaks to those with access to the keys. As opposed to a normal HTTPS service where only transport security is achieved and the data would be plaintext after reaching the organisation. GlobaLeaks for example allows the whistleblower to pick specifically which journalists should receive the leak, and encrypts it using those specific journalists' own PGP keys. In case the server is compromised then the data is still protected by that PGP encryption which only the specific journalists is able to decrypt.

6.4 Clientside Cryptography

Use client side cryptography to increase transparency further. Some newspapers already make use of PGP for storing their leaks. A lot of them perform this encryption on the server side and others do it partially on the client side. Moving the process to the client side increases transparency, by letting clients observe it. This addresses the problem wherein server administrators can access the plaintext content of a leak.

Journalists should make sure to publish their public keys, so that whistleblowers have the opportunity to encrypt leaks manually if they want to. Any client using cryptography needs to be open source so that whistleblowers or "good samaritans" can also verify that the client works correctly with the correct keys.

One major problem with clientside cryptography on websites is that the JavaScript can be easily modified by the server without detection, since the server sends the client

code to the user. In order for clientside cryptography to be trustworthy from serverside backdoors there must be some system to verify the integrity of the client.

However, even without the extra integrity on the served javascript, if the server can be trusted to not modify the script then the inclusion of clientside cryptography means that the server does not have to worry about encrypting the leaks itself. For example, both SecureDrop and GlobaLeaks do a lot of work to make sure that the original plaintext sent and received by the webservice is never saved on disk or in memory unencrypted.

It must also be noted that any file contents must be encrypted too, since the submitted files may contain data which can identify the whistleblower. The solutions deployed by DN and Sveriges Television for example do not employ this functionality properly.

This thesis will look more closely at using javascript cryptography and try to solve the problems with the script integrity when the server itself that is hosting the script can not be fully trusted. The next chapters go more into detail with this, proposing a new design for a system which uses javascript cryptography.

6.5 Avoid Third Party Content

The whistleblower website can not contain content from any third party services since that would leak metadata about usage of the site. Even worse, if scripted content is hosted from a third party service it could present a security vulnerability if the third party service is malicious. In the case of a website, javascript, stylesheets or other dynamic content should not be included from a third party domain. Images and other static content is also not recommended since that could leak usage data to those services. In general, avoid any third party content on the website.

6.6 HTTPS Everywhere

If the service is built as a website, then it must be served over HTTPS in order to maintain confidentiality and integrity. If the whistleblowing service is hosted on the same domain as the normal content, then it is also preferable if the server uses HTTPS across all parts of the site. This effectively increases the anonymity set of users connecting via HTTPS to the website to equal the number of readers of the newspaper. Otherwise there could also be a risk that the few users connecting over HTTPS are only there for the whistleblowing.

Chapter 7

New Design

This section proposes a new design for a web based whistleblowing system that aims to improve on the current solutions. The main new features being clientside cryptography with tamper detection and stronger journalist anonymity by use of key-private cryptography. The purpose of the design is to explore the feasibility of such features in the current state of technology, and if they can be used to further improve on state-of-the-art solutions.

The system itself consist of four applications: A whistleblower client, a web service, a client application running on a journalist's computer, and a shared source code repository, which contains the signed source code for the whistleblower client. Figure 7.1 shows an overview of the system I am proposing.

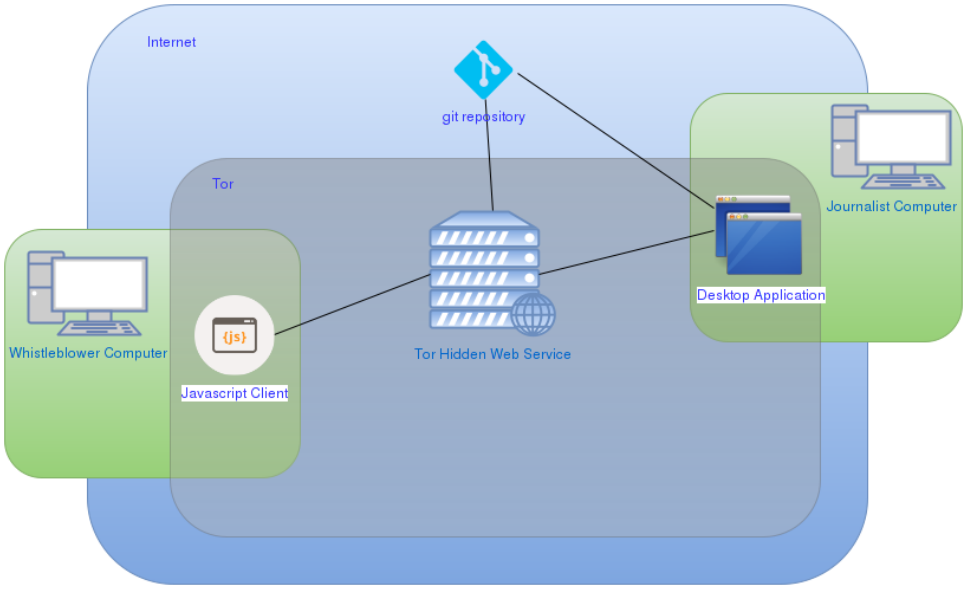

Figure 7.1. Overview of the system

In order to support clientside cryptography in a web application, the whistleblower has to obtain the client application so as to run the cryptographic operations on the whistleblower computer. To do this the whistleblower's browser instance first has to fetch the client application from the webservice. This client is what interacts with the system on behalf of the whistleblower and what implements the submission protocols described later in this chapter. Separating the cryptographic parts of the system from the server differentiates this design from existing whistleblowing platforms. Those platforms instead run cryptographic operations on the server after receiving these in plaintext from the whistleblower computer. In this system the server hosts a javascript client which is downloaded and does the encryption before submission.

In order to prevent basic attacks against the integrity of this client by the server administrator, e.g. by implementing backdoors in the encryption routines, a verification protocol is implemented by the journalist client. This verification protocol is described in the protocols section of the chapter.

The system has two types that encapsulate interactions that occur between whistleblower and journalist. The first type is the *submission* type, which is the initial first request made by a whistleblower to one or more journalists. It is referred to as a submission or a leak. The second type is the *response* type, which is the ensuing response to the submission type and contains the back-and-forth conversation between journalist and whistleblower.

The system also looks at features which further improve anonymity and privacy for both whistleblowers and journalists. By using cover traffic protocols it obfuscates metadata about timing and activity based on the inputs. The two protocols regarding the submission and response types are both emulated with dummy inputs to make it more difficult to determine whistleblower timing. Two dummy protocols are described later in this chapter.

The new features added in this system could be implemented into the existing projects SecureDrop or GlobaLeaks, but would require a substantial amount of code rewrite since both solutions rely a lot on serverside security. Such a rewrite may in the end resemble something completely new. That said, the existing serverside security features such as intrusion detection systems, memory protection and sandboxing features should be reused. These features are generally not coupled with the whistleblowing platforms themselves but with the operating system that runs them.

To be clear, the new features that are not implemented by SecureDrop or GlobaLeaks that this system proposes are the following:

1. Key-private clientside encryption (section 7.2)
2. Integrity verification of clientside script (section 7.4.1)
3. Metadata obfuscation by use of dummy cover traffic (section 7.4.2, 7.4.3)

The system described in this thesis also includes a desktop application as the journalist client, which is not included in either GlobaLeaks or SecureDrop. Both projects instead opt for a web interface and let journalists handle the encryption and decryption

by themselves. Due to the automatic polling used by the verification and dummy protocols, I believe it is necessary for the system described in this thesis to provide a client for journalists as it would quickly become too tedious for a journalist to do these operations by hand.

7.1 Security Policies

In the context of this system, three different types of users exist: whistleblowers, server administrators and journalists. The user types interact around two different types of objects in the system: submissions and reply threads. A submission is a message made to one or more journalist. A reply thread is a conversation connected to a submission, it contains responses that can be made back-and-forth between journalist and whistleblower. There are four applications which different users are allowed to modify or use: the whistleblower client, the webservice, the journalist client and the source code repository for the whistleblower client. The following policies describe which type of user may do what in the system.

- Anyone can submit new submissions into the system. Only journalists are valid targets for the submissions. Only the journalists to which the specific submission was addressed to can read that submission.
- Any read or write interaction of the plaintext reply threads can only be made by either the original whistleblower or a journalist that was part of the initial submission.
- Whistleblower client is saved in a shared source code repository. Anyone who changes the code must sign with their private key. Only journalists or server administrators are authorised to change the client code. Unauthorised changes to the source code are detected.
- Whistleblowers are anonymous when they connect to the service.
- Journalists fetching submissions are anonymous within the group of journalists. Journalists providing dummy cover traffic have the same level of anonymity against the server as whistleblowers.
- For transparency, all application code for this system should be readable by anyone. Code submitted should be reviewed for errors and security holes.

7.2 Submission Protocol

Figure 7.2 shows interaction between whistleblower, server and journalist, when a whistleblower wishes to initially submit a leak to one or more journalist.

When a whistleblower submits a new leak it creates a plaintext packet m containing the input files and information that the whistleblower wishes to submit to the journalist(s). In addition to

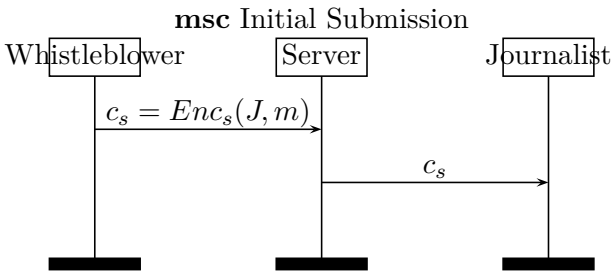

Figure 7.2. Enc_s is an key-private [19], asymmetric and robust [21] encryption scheme. m is the packet of information that the whistleblower wishes to send to the journalists. m also contains a random identifier $m.id$ and a secret key $m.secretKey$. The secret key $m.secretKey$ is to be used for a symmetric authenticated encryption scheme (Enc_r). J is the set of journalists selected to receive the message from the whistleblower. c is the resulting ciphertext.

the contents of the leak itself, a new random identifier $m.id$ and a new secret key $m.secretKey$ are also created and attached to the packet m. The $m.id$ will be used to later identify the leak, and the $m.secretKey$ will be used for the Enc_r encryption scheme used for replies. $m.id$ and $m.secretKey$ are displayed to the whistleblower, who is asked to save this information. m is then encrypted by an encryption scheme Enc_s using selected journalists public keys J. Enc_s outputs the ciphertext c_s which is what is actually submitted to the server. The s subscript is used to denote submission, while the r subscript is used to denote reply.

Upon receiving the submission from the whistleblower, the server does not notify the journalists. Instead the journalists periodically check the server for new content. This connection is made over Tor, to prevent an outside observer from being able to determine when the server received new leaks based on any form of notification coming from the server to the journalists.

Upon the periodic check, any new ciphertext c is downloaded. Enc_s is key-private [19] and does not include any hints as to which journalist it is for. This way the journalists can not determine if the message was for them without trying to decrypt the packet. By doing this, each journalist has to download every message. Thus the server has no way to determine which journalists the messages were for, because every journalist interacted with the server the exact same way. Neither by ciphertext, because the encryption is key-private, nor traffic analysis based on who downloaded the message, can it be determined who is the recipient of the message.

By using this periodic check, the system achieves anonymity between journalists. The server knows that a journalist has downloaded the message, but it does not know which journalist the message was actually meant for. The key-private encryption Enc_s makes it more difficult for an adversarial server from removing messages directed to specific journalists.

However, at least three problems remain:

1. The client code sent by the server could be trivially tampered with to bypass any encryption.
2. A server administrator could still determine the timing of when an actual leak was submitted to the server by simply seeing when the submission is received.
3. An outside observer could also determine the timing of the leaks. E.g. the ISP running the network that the journalist connects via could correlate Tor traffic activity to a new leak being downloaded from the server.

We leave these problems to be addressed by *Dummy and Verification Protocols*, described later (section 7.4) in this chapter.

7.3 Replies to the Submission

One or more journalists may choose to reply to the whistleblower's submission. This is where the reply functionality of the system comes in. The reply functionality works with a reply thread object. The reply thread contains an ordered list of messages sent between journalists and whistleblower. The whistleblower client earlier created a random identifier $m.id$ and a secret key $m.secretKey$ which were included in the initial submission. These are used to identify, authenticate and encrypt the reply thread objects that are associated with that submission. $m.id$ identifies and authenticates access by the server to the encrypted object. It is simply the filename of the object on the server. $m.secretKey$ encrypts the object, and thus limits access to the actual plaintext contents of the object. In order to successfully access the reply thread, a user would need both $m.id$ and $m.secretKey$.

Each read and write operation on reply threads are joined into one in order to achieve indistinguishability. This is so that analysis can not determine what type of interaction was had with the reply thread, if the read and writes were distinguishable from one another, then that may give away timing information about either the whistleblower or journalists working with it. To do this a policy enforced where any read must be followed by a write, and any write must be preceeded by a read. In addition to joining reads and writes, the encryption scheme Enc_r adds padding to ensure the ciphertext $cr_{m.id}$ size stays the same even when new messages are added to the thread. Otherwise the server could observe the size changing to indicate write activity on the reply thread.

When first replying to the whistleblower submission, the journalist initializes a new reply thread object $r_{m.id}$ containing an internal list of messages. The journalist appends their message

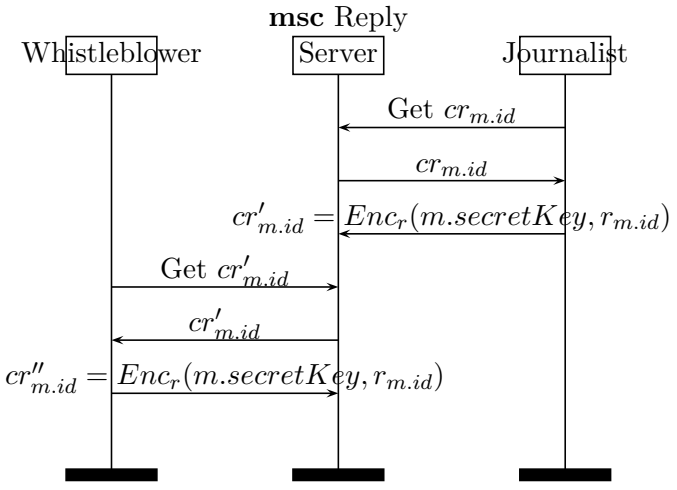

Figure 7.3. Enc_r is a symmetric authenticated encryption scheme with a large fixed size padding. $r_{m.id}$ is the reply thread, identified by $m.id$. $cr_{m.id}$ is the encrypted reply thread.

to the list, encrypts the reply thread object using Enc_r and uploads it to the server. Any subsequent interactions decrypt the encrypted reply thread object $cr_{m.id}$ to get $r_{m.id}$, append new messages to the internal list of messages, re-encrypt the reply thread object into a new $cr_{m.id}$ and uploads the thread again to the server.

New messages to a reply thread can be detected by the journalist client by automatic polling of each thread. The polling mechanism performs the same read and write protocol described earlier in order to remain indistinguishable.

The server tracks and authenticates reply threads using associated $m.id$ that was included in the initial submission. However, it can not determine which journalist or whistleblower just interacted with the thread or if the interaction was a read, a write or both. The server also can not determine which submission object was related to the reply thread. The server allows anyone to write or read the encrypted reply thread object, so long as they present the identifier $m.id$. For this purpose $m.id$ must be picked randomly and with enough entropy for it to be secure against bruteforce attacks from outsiders.

By using an authenticated encryption scheme, the server can not try to replace the reply thread with junk data without the involved parties detecting the tamper when decrypting as the authentication would then fail. Because of the secret $m.id$ only the server administrator, the whistleblower and the involved journalists should be able to write on the reply thread. Therefore a failed decryption should alert the involved parties that possibly the server is misbehaving.

The padding size for the reply thread object determines how many messages can fit into the object. Even so, assuming that the average message is 256 characters long, a 1 MB file could fit almost 1000 messages in one object. New writes could still cause the reply thread object to grow past the padding size. When it does, the object could simply increase the padding further, for example by doubling the padding size. Alternatively the object could be recycled by removing old irrelevant messages.

Similar to the submission protocol, the reply protocol also has problems.

1. The server administrator could drop incoming writes without the subjects of the reply thread detecting it.
2. An observer could also correlate the timing and size of traffic with the service to determine when a subject interacts with a reply thread.

These problems are addressed by *reply cover traffic* described later in this chapter.

7.4 Dummy and Verification Protocols

Three extra protocols are added to specifically counter threats against the whistleblower's anonymity and the integrity of the whistleblower client. The first verification protocol attempts to verify the integrity of the whistleblower client. The second and third protocols attempt to anonymize whistleblower and journalist acitivity on submissions and replies by introducing dummy interactions which work as cover traffic for real interactions.

7.4.1 Client Integrity Verification

Since the server provides the whistleblower client, it can also easily modify that client for malicious purposes. In order to prevent this, the journalist client or a standalone verifier will periodically perform a random sample verification against the served client.

The protocol relies on two assumptions. First, the verifier's interactions with the server can be made indistinguishable from those made by the whistleblower client. Second, the full client

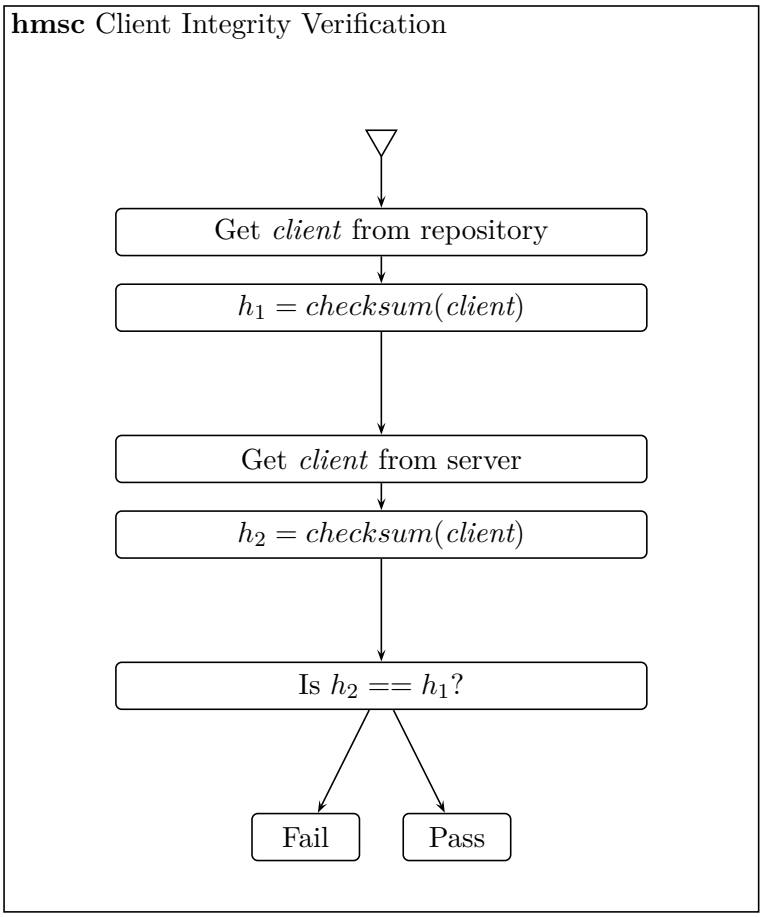

Figure 7.4. Client Integrity Verification

is fetched from the server all at once. The client can not be changed by the server after the initial request. Any subsequent requests do not result in any server-injected code executing on the client's computer.

The verification also relies on there being a shared repository for the whistleblower client. The contents of this repository should be cryptographically signed and any interactions with the repository must verify these signatures. The key-trust infrastructure is assumed to already be in place, e.g. the organisation which hosts the whistleblowing service have had all involved members exchange their public keys.

Figure 7.4 shows the algorithm for verifying the client.

The verifier starts by fetching the latest client from the shared repository, verifying that the repository code is signed and trusted. It then computes a secure checksum h_1 of the client data. For this purpose a cryptographic hash function that is collision resistant and irreversible is suitable. Some time after computing h_1 it fetches the client from the whistleblowing server. This fetching action is done in a way that is indistinguishable for the server to tell whether the request was made by a whistleblower or verifier. Then it computes a hash h_2 of the client data found on the server and compares h_1 and h_2. If h_1 and h_2 are the same, the verification passes.

Assuming this verification is run randomly and periodically, for an arbitary incoming request the server has a chance relative to the ratio between the number of requests made by the verifier and the number of requests made by the whistleblower to try to maliciously modify the client data without being detected.

$$p_{detection} = \frac{V_r}{V_r + W_r} \tag{7.1}$$

where V_r is the number of requests coming from verifiers and W_r is the number of requests coming from whistleblowers. If $V_r > W_r$, then the malicious server is likelier to be caught by the verifier than not.

To prevent an attack where the requests to the code repository are observed and correlated to determine if an incoming client request is a verifier, the code repository pulls should be made from a local repository. This local repository should then be updated from the master remote repository on a regular basis, independent from the timing of a verification request. In the case where a local repository is out of date and a checksum mismatch happens, the verifier could force an update of the local repository then compare again against the new latest version. If the checksum of the server-fetched client still does not match, then the verifier can signal that the client is being tampered with.

Assuming there is a bias in real whistleblower requests to e.g. occur during the day, there will be a non-random distribution of which hours during the day that whistleblowers are active. In a worst case scenario, all whistleblower could for example submit at the exact same time during a specific time of the day, which would make the distribution of real whistleblower requests easy to predict and thus more open to attacks. Adding random cover traffic will however add entropy to the combined distribution. The larger the cover traffic is in comparison to the real traffic, the less significant the real traffic becomes when performing analysis on all requests.

The integrity verification protocol works with the probabilty $p_{detection}$ that it detects tamper of the client protocol. By setting a $p_{integrity}$ as the accepted minimum probability that the protocol detects tamper, we can calculate an estimate of the rate of supported whistleblower requests that are covered by the integrity verification protocol. The following equation shows supported whistleblower requests as a function of the number of requests coming from verifiers V_r.

$$w_{supported} = V_r(1 - p_{accepted}) \tag{7.2}$$

Using this equation we can adjust how many dummy requests are made based on how many whistleblowers we expect and a detection probability that is deemed acceptable.

Further we can also calculate the following inequality, based on the rate of dummy requests and the detection probability. So long as the equality is satisfied the accepted probability is fulfilled. The protocol will then detect tamper with at least that probability.

$$V_r \geq W_r/(1 - p_{accepted}) \tag{7.3}$$

The integrity verification protocol is assumed to be included as part of both the Submission Cover Traffic (7.4.2) and the Reply Cover Traffic 7.4.3 which will both fetch the whistleblower client as part of their respective protocols.

7.4.2 Submission Cover Traffic

In order to verify that the server behaves correctly and does not block submissions from whistleblowers, the same principle is used as when verifying the client integrity. Again it relies on the

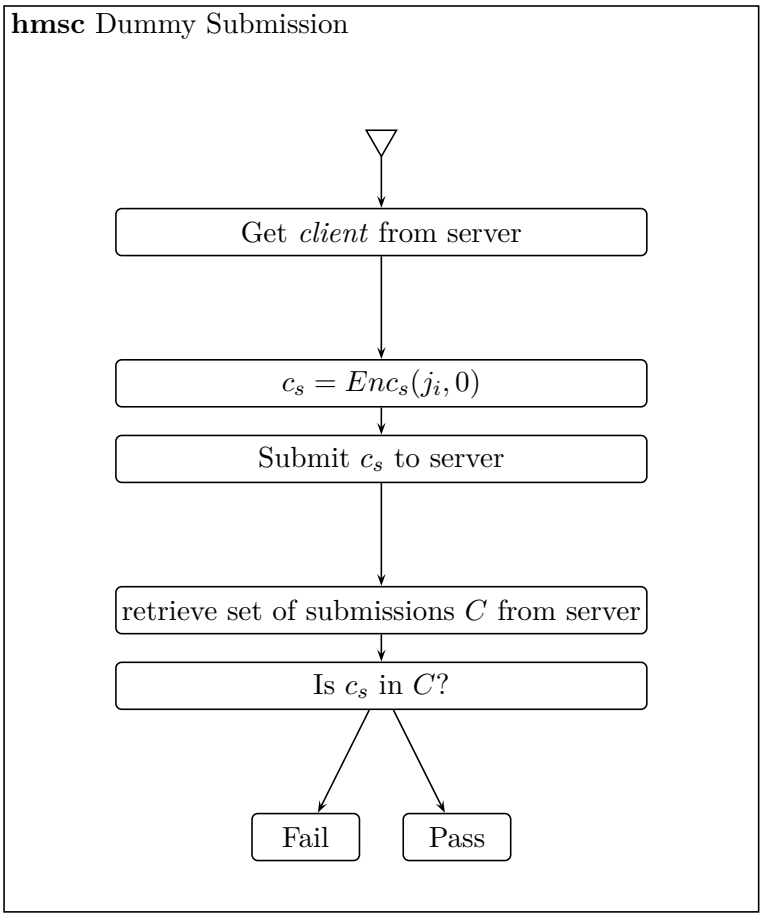

Figure 7.5. Dummy submission. Enc_s is the same encryption scheme used for submissions. c_s is the encrypted submission object. C is the set of submissions stored on the server.

assumption that the server can not distinguish from a request made by the verifier and a request made by a whistleblower.

The cover traffic protocol adds dummy submissions to mix with the normal submissions. The purpose of this is to further hide metadata around timing and ensure that the server is accepting new submissions. The timing metadata is for example the point in time when a submission was made. This data could reveal information about the whistleblower's time zone and habits.

Figure 7.5 shows the algorithm for a dummy submission. This algorithm is repeated randomly over time to form the cover traffic for submissions.

The verifier fetches the whistleblower client. Then after t_1 seconds it creates a dummy submission encrypted using the journalist client's public key and containing data that can be identified as dummy data. It submits this dummy submission to the server. When the next polling for the submission protocol occurs, the client also looks for its submitted dummy submission. If it does not find the dummy submission or the decrypted submission does not match the unencrypted content, then it can signal that the dummy submission failed.

By using key-private trial-and-error decryption where each journalist is required to retrieve

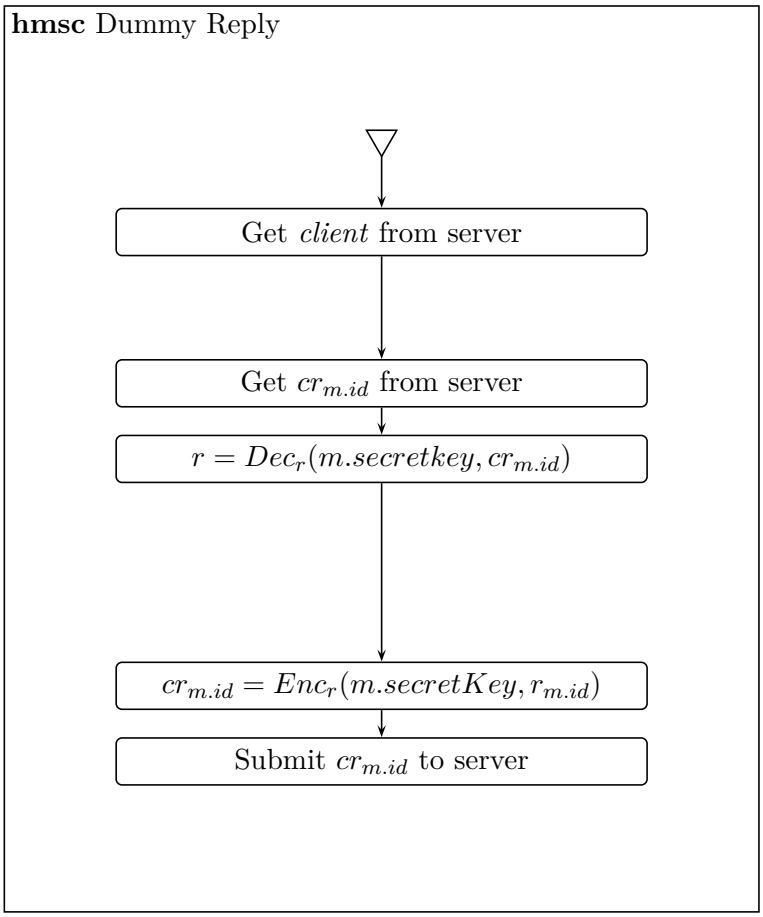

Figure 7.6. Dummy Reply. Enc_r is the authenticated symmetric encryption scheme used to encrypt reply threads. Dec_r is the decryption function for the same scheme.

every message, and since each receiver also generates the cover traffic which is sent around to itself, the system falls into the category of *receiver-bound cover* [26]. Meaning that it should be more difficult to perform statistical attacks.

t_1 is the time between the initial request and the submission of the dummy submission to the server. d_s is the size of the submission data. These two variables have to be set randomly in order to make the dummy submissions indistinguishable from real submissions. To do this the variables should be picked randomly from a minimum and maximum bounds.

7.4.3 Reply Cover Traffic

The reply threads can be similarily covered with dummy replies to hide timing metadata. The reply cover traffic works the same way as the normal reply thread traffic, by reading the file, re-encrypting it, and then after t_{rand} seconds re-uploading it.

With the enforced read-then-write behaviour for interacting with the reply functionality, there will be a time difference t_δ between the read- and the write request. This t_δ would have to

be randomized, or an adversary would be able to distinguish verifier from whistleblower, or the type of interaction that was had. For example, sending a write immediately after a read could be seen as too quick to be written by a human. At the same time, sending a write a very long time after a read could indicate a write.

If each journalist client provides continual cover traffic for replies only they have knowledge of, then the server may be able to determine the number of journalists that are interacting with a certain pseudonym by simply counting the number of requests. The mean number of requests over time to a certain reply file would be proportional to the number of journalists connected to it. To counteract this the journalist clients scale their individual rates to go slower by increasing the maximum bounds of the random function. This calculation could be determined beforehand, based on estimations of how many whistleblowers will connect to the service.

If each reply file has to be covered to a certain rate, then it will also quickly become a scalability problem for the journalist clients and server to be able to keep those rates up.

The cover traffic should be implemented as part of the polling mechanism. The mechanism should then occur at a set randomly distributed rate f_{reply}. In order to cover all reply threads, the dummy requests have to be distributed across the journalists assigned threads. Depending on this rate and the number of reply threads, the frequency of polling and cover traffic on one specific reply thread can be calculated as follows.

$$F = f_{reply}/n_{reply} \tag{7.4}$$

where n_{reply} is the number of reply threads. This frequency will later be used to determine if the amount of cover traffic is enough to protect a reply thread object.

To detect if the server does not drop packets, an additional field can be added to the reply thread object that adds a validation field for each journalist client. Whenever doing a request, dummy or not, the journalist client can set this field to a new token. On the next read, the client can then verify that the token is still there. The server should then also implement a concurrency control to prevent race conditions happening between different clients trying to write on the object at the same time.

The concurrency control has to be done in a way so as to not enable the server to use the concurrency control itself to drop packets. One way of doing this could be to implement a check on the journalists' client, which does an extra read after the write. Then it checks if either it's verification token is the same or if there is a new token from another journalist. If neither is true, then the client alerts the verifying journalist that something is wrong.

7.5 Calculating Storage and Data Transfer

To calculate the storage and data transfer requirements of the system, a number of constant settings are required. $p_{integrity}$, $p_{submission}$ and p_{reply} are the accepted probabilities that tamper should be detected for the client, submissions or reply threads. To make estimations easier, these are sometimes assumed to be the same value and set as p. $d_{submission}$ is the set average size of a submission. d_{reply} is the set size for a reply thread object. f is the rate of indistinguishable dummy requests made per hour by one journalist. The dummy request rate can also be split into rates specific to the type of interaction that it is covering, i.e. $f_{submission}$ is the rate of dummy requests made to cover submissions and f_{reply} is the rate of dummy requests made to cover reply threads. j is the number of journalists connected to the system. w is the number of whistleblowers connected to the system for the time period. $w_{submission}$ is the rate of whistleblower requests where the whistleblower makes a new submission to the system. w_{reply} is the rate of whistleblower requests where the whistleblower interacts with a reply thread.

The size of the metadata for the networking packets (e.g. HTTP headers) are considered neglible and are not included in the estimations.

7.5.1 Storage

The total required storage s_{total} includes submissions and replies made by both real whistleblowers and dummies.

The following equation shows the storage required by dummies. Only the dummy submissions are stored on the server. New dummy reply threads are not created, thus only submissions affect the storage requirements.

$$s_{dummies} = f_{submission} \cdot j \cdot d_{submission} \tag{7.5}$$

The following equation estimates the number of real submissions and replies. It is assumed that for every submission there is a reply, and every whistleblower makes a submission and therefore a reply. The system only allows for max one reply thread per submission, meaning there are never more replies than submissions.

$$s_{real} = w_{submission} \cdot d_{submission} + w_{reply} \cdot d_{reply} \tag{7.6}$$

Finally the total storage estimate is the sum of $s_{dummies}$ and s_{real}.

$$s_{total} = f_{submission} \cdot j \cdot d_{submission} + w_{submission} \cdot d_{submission} + w_{reply} \cdot d_{reply} \tag{7.7}$$

7.5.2 Data Transfer

The total required data transfer t_{total} for real and dummy interactions with the system is estimated under the assumptions that each submission and reply is targeted to every journalist. This is a worst case assumption, as reply object would only be fetched by a few of the journalists otherwise.

The required data transfer for dummy submission requests is estimated by the following equation. Each journalist posts f new submissions which each other journalist downloads.

$$t_{dummy} = f_{submission} \cdot j^2 \cdot d_{submission} \tag{7.8}$$

The data transfer required for requests from whistleblowers is calculated as follows. This is under the assumption that each submission is targeted to every journalist.

$$t_{submission} = j \cdot (w_{submission} \cdot d_{submission} + w_{reply} \cdot d_{reply}) \tag{7.9}$$

Data transfer for reply threads is constant at the rate f_{reply} per journalist. The data transfer from replies coming from whistleblowers is assumed to be one per whistleblower visit.

$$t_{reply} = f_{reply} \cdot j \cdot d_{reply} \tag{7.10}$$

Finally, the total required data transfer is calculated as the sum of the previous estimates.

$$t_{total} = j \cdot (w_{submission} \cdot d_{submission} + w_{reply} \cdot d_{reply}) + f_{submission} \cdot j^2 \cdot d_{submission} + f_{reply} \cdot j \cdot d_{reply} \tag{7.11}$$

7.6 Evaluation of Security Policies

The encryption of submissions using the journalists public key as part of Enc_s enforces the first policy, so long as Enc_s is secure. The whistleblower client only has journalist keys as valid recipients in the upload interface.

Reply threads are encrypted by Enc_r with a key that is distributed in the original encrypted submission. Enc_r is authenticated, meaning that only those that have the key are able to create a valid ciphertext. Reply thread access is also limited by knowledge of a random identifier. So long as the identifier is very difficult to guess anyone except the journalists and the whistleblower involved are able to upload valid ciphertext to the server. The server can still in some sense interact with the ciphertext, e.g. by replacing it with junk, but it can not produce a valid ciphertext.

The client itself is secured by the integrity verification protocol, which should detect if changes are made to alter it by the server. Observers, such as the outside observer and the authority, are not able to alter the client so long as the Tor hidden service protocol is secure (similar to TLS).

Whistleblower anonymity is dependent on the anonymity of Tor. There is also further anonymity in timing metadata by the use of dummy submissions. Journalists are anonymous within themselves due to the continuous automatic polling of all incoming submissions. Dummy cover traffic is to be made as indistinguishable from the whistleblower traffic as possible.

To fulfill the last policy requiring transparency, I recommend making all code publicly available as an open source project. Setting up a routine of code reviews and penetration tests would further help the security.

Chapter 8

Prototype Implementation

This chapter describes the prototype implementation part of the thesis. This chapter contains a section on the planned features of the full system, then a section on the different prototypes that were implemented. Finally a section with some results on performance and some estimations for the usage of the system.

8.1 Features

This section lists different features which should be implemented in a full version of the system.

Website based interface for whistleblowers

The whistleblowing service should be built as a website. All users of current modern operating systems have access to a web browser and are likely familiar with using websites. I believe this is the most accessible way for a user today to interact with a system.

Submission with file upload feature

The basic functionality of a whistleblowing service is a form which allows whistleblowers to submit one or more files as well as some text information. The service is as such required to support upload of multiple files and some text information.

Selectable journalist receivers in submission interface

Having a control to allow the whistleblower to select which journalists are to receive the leak they want to submit gives the whistleblower more control over which journalist or organisation they wish to talk to, as well as limits the number of people who can see the text.

Server file storage

The content submitted must be stored somewhere until the intended recipient has received them. For this purpose there has to be somewhere to store files. These files should be made accessible for journalists. In order to support dummy submissions fully, this functionality must be able to store several gigabytes and several thousands of files.

Authenticated leak API for journalists

New submissions should be made available to the receivers through an API. Access to the API should be restricted only to those that are authenticated: either the server administrator or the journalists. Only server administrator and journalists should be able to see the size of a leak or when it was submitted.

Authentication for journalists should be anonymous. One easy way of accomplishing this is to use a shared authentication token that only the journalists and server knows.

Anonymous asymetric encryption for submissions, Enc_s.

Each journalist should have a keypair for an anonymous asymetric encryption scheme. An example of an anonymous asymetric encryption scheme could be to use El-gamal or Cramer-Shoup (optionally in an elliptic curve) as key encapsulation methods for AES. The assymetric scheme encapsulates a symmetric key that encrypts the contents of the submission.

The submissions that are sent to the service should be encrypted using this scheme. This is to enforce the required confidentiality where only journalists are allowed to view the leaks.

Tor Hidden Service

The whistleblowing service should be made available as a Tor hidden service. This makes it more difficult for adversaries to monitor network traffic going to and from the service, as well as make it harder to seize the physical server. Hosting a Tor hidden service generally provides a good amount of anonymity for the server itself. If the service is only made available via Tor then that promotes whistleblowers to use Tor, which provides strong anonymity and privacy for whistleblowers. All interactions with the server should be made via it's Tor hidden service in order to anonymize both whistleblowers, and to an extent journalists.

Connecting over Tor to a Tor hidden service also automatically provides similar security like that of TLS for normal HTTPS websites.

Javascript Client for submission encryption

The whistleblower client should be implemented in javascript. This allows whistleblowers to easily get the client from the webserver, because it will load directly in their browser. Javascript is also supported by the Tor browser, unlike other types of client script such as Flash or Java.

New submissions should be encrypted with the journalists' keys before being sent to the server. This makes it impossible to passively eavesdrop on leaks, even if the adversary has access to the server hosting the website. Any such adversary would then have to change the javascript that is executed in order to gain access to the plaintext. The honest server would never have to worry that the submission is left in plaintext in memory or on disk.

The client should be able to support the submission and reply thread protocols described in chapter 7.

URL/DNS leak-free contact information

The contact information regarding how to initially contact the service should be obfuscated to prevent URL and DNS leaks from occuring. This can be done by hosting the how-to in a hidden element on the main newspaper that is accessible via a javascript popup. This assumes that everything is served over HTTPS or with equivalent integrity, so as to prevent basic man-in-the-middle attacks.

Client Integrity Checker

To prevent man-in-the-middle attacks on the javascript client by someone in control of the server there has to be a way to verify the integrity of the javascript. This can be done by comparing checksums of the served content to expected content. This client should implement the Integrity Verification protocol described earlier in chapter 7.

Code Repository with Signed Commits

For the clientside integrity checker to work, there must be a shared code repository which supports signed commits. The Git source code management system has the functionality to allow for PGP-signed commits.

Journalist Client

The journalists should have access to a specialized client to make the process easier for them. This client will communicate with the web service and handle the cryptographic operations for the journalists. This may include key generation, decryption, verification and connecting via Tor to the hidden service. These tasks are otherwise difficult for non-technical people to perform, so automating them makes it less likely that a journalist will slip up in operational security and lets them focus on the journalistic work.

This client will automatically poll for new submissions and send new replies to the server. The polling will be done periodically and all submissions will be downloaded and decrypted in order to provide anonymity between the journalists.

This client should implement the Submission and Reply Thread protocols described earlier in chapter 7.

Dummy Submission Verifier

The dummy submissions protocol must be implemented to prevent against server censorship attacks against specific journalists. In order for this to work, the verifier must be run with the same journalist key that it is trying to protect. Therefore the dummy submission verifier should be part of the the journalist client.

This verifier should implement the Dummy Submission protocol described earlier in chapter 7.

Reply thread Encryption, Enc_r

Reply threads are encrypted with the authenticated symmetric encryption scheme Enc_r. Enc_r has to be an encryption scheme that has extra padding to hide the actual length of the content and to make it difficult to distinguish reads from writes. One way to accomplish this in practice is to use AES/CCM. Padding can be added inside the plaintext as a large chunk of random data, aligned so that when encrypted the ciphertext always ends up the with same length.

Dummy Reply Verifier

The reply cover traffic protocol should be implemented to further anonymize timing metadata around activity on reply thread objects. It should also help with detecting server censorship attacks on reply threads. Like with the dummy submission verifier, the reply verifier should be implemented in the journalist client.

This verifier should implement the Dummy Reply protocol described earlier in chapter 7.

8.2 Prototypes

The system was implemented as a series of prototypes. A whistleblower client, a backend server and a journalist client. The purpose of these prototypes was to see the applicability of the system.

All of the source code for the prototypes will be made publicly available with the publication of this thesis, in order to show reproducibility. The source code will be available on this website: `https://github.com/Tethik/whistleblower`.

Whistleblower Client

The whistleblower client was created in Javascript and hosted on a Tor hidden server. The client used the OpenPGP.js and the SJCL.js libraries to implement a PGP encryption on the submissions and a 128-bit AES/CCM encryption on the reply threads. While the PGP encryption does not achieve the required key-privacy property, it has similar behaviour when determining performance. The wanted ECC Elgamal encryption scheme was not implemented due to time constraints. The client was hosted in a git repository and deployed only if the latest commits had a valid signature trusted by the local PGP keystore.

The client otherwise supported the required features of an encrypted multiple file upload submission and the reply thread functionality for returning whistleblowers.

Backend Server

The backend server was implemented in Python using the Flask web framework. Requests were routed from a Tor hidden service which pointed to a locally hosted web service. Functionality wise, the backend server provided the submission and reply thread API. Reply threads and submissions are stored as files in the local filesystem. It also hosted the whistleblower javascript client, which is automatically deployed from a locally hosted git repository. Each checkout from the git server was verified by a PGP-signature based on installed trusted keys in a local PGP installation.

Journalist Client

The prototype journalist client was implemented in Java as a GUI application. JavaFX was chosen as the GUI framework and API requests to the server are routed over a Tor SOCKS proxy. Cryptography is handled by the BouncyCastle framework. The client runs using an internal PGP keystore, generating a keypair for the journalist to use. Both submission and reply thread functionality were implemented. The client automatically polled for new submissions in the background, and could decrypt incoming submissions. Reply threads could be created and a back-and-forth communication worked with the whistleblower client.

Verifiers

Verification and dummy protocols were not implemented into the journalist client due to time constraints. However, the integrity verification was prototyped as a standalone script.

Selenium [45] is a framework which allows programmers to programmatically control a web-browser through code, typically used for unit testing websites for bugs and ensuring that different functions still work. Using Selenium to control a Tor Browser, the same browser most likely used by Tor users and whistleblowers, the requests sent by the verifier become identical to those sent

by a real user. Through this method, I hoped to achieve indistinguishability between dummy and whistleblower requests.

Assuming this method works, it can also be used to send data and even run javascript. So it should be possible to do the dummy submissions and replies via emulation in the Tor-Browser Selenium instance.

8.3 Performance

This section looks at the performance requirements of the system. First the performance of Tor, specifically what rate of dummy requests can be accomplished. After that the thesis looks at how indistinguishable requests made by Selenium are from those made by a human user. Finally the storage capacity and data transmission needed is estimated based on how many whistleblowers and journalists would use the system.

8.3.1 Performance of Tor

In order to determine the feasibility of providing different types of cover traffic over Tor, a simple test was performed to measure how often HTTP request could be made to a hidden service. The test was done using the default settings for Tor connecting to a hidden service running on the same computer. Each new connection would be forced to use a new circuit. The test was created using the python framework Stem which communicates with the backend Tor process.

Through this simple test, a new-circuit HTTP request was possible on an average of every 10 seconds. Likely this 10 second delay is a limit coded directly into the Tor application to prevent abuse by clients that want to slow down the network. It is possible that by interacting directly with the Tor protocol this delay can be reduced further, but that was deemed to be outside the scope of this thesis.

When using the Tor browser the default time-out until Tor chooses new circuits is every 10 minutes. Assuming that Tor browser is also used for providing cover traffic, the aforementioned time-out might not be configurable. This would mean that the cover traffic has a minumum bounds for it's random timings between tests of 10 mins per instance running. If the 10 minute miniumum bounds is too long in between requests, one could alter it down to at least 10 seconds, but it would require extra configuration or implementation, which might make the system less practical.

If the minimum bounds is set to 10 minutes and maximum to 20 minutes, the rate would be four requests per hour. This rate applies for a single client and could scale up by adding more clients.

$$f_{low} = 4j \tag{8.1}$$

If the maximum bounds are set to 20 seconds, the average rate of requests would be one request every 15 seconds, or 240 requests per hour.

$$f_{high} = 240j \tag{8.2}$$

8.3.2 Indistinguishability between Requests

The different verification protocols rely on the assumption that dummy and whistleblower requests made to the server are indistinguishable. In practice, this can be accomplished by emulating requests coming through the Tor browser. Since most web users connecting over Tor are

using the same browser we can create request that look the same as that browser. Browser can be programmatically operated by testing frameworks, one such framework being Selenium [45].

Through testing, raw GET-request data from a Selenium controlled- and a human controlled browser were captured with a network sniffer and then compared for differences. Data sent by the two methods were identical.

There was however a slight difference in the behaviour of the verifier script versus the human user which lead to extra requests being made by the Selenium browser. This was likely because of how the verifier script opened every linked resource separately to download and verify the data. The resource request would be fetched from the browsers cache and thus not trigger a request to the server, however as a side effect the browser would still request the server's icon image.

A workaround to this difference in behaviour could be to either alter the verifier script to find another way to save the requested data. Another way could be to compile the whole whistleblower client in one singular HTML file, which would not require extra requests to fetch. A more efficient approach would be to try and copy the behaviour that the Tor Browser exhibits, by copying the client headers and parsing the HTML the same way. Since the Tor Browser is open source, it is also possible that that code could be repurposed for the verifier.

8.3.3 Storage Capacity and Data Transmission

Using the Tor new-identity request rates f_{high} and f_{low} measured in section 8.3.1, we can now make some estimations on the needed storage capacity, data transfer and the number of supported whistleblowers. These in turn can be used to evaluate the system to see if it would be practical in a real world scenario.

According to a report by International Transparency Italia, they had a total of 124 whistleblowers [46] contact them within in a one year period. Another report by the U.S. Securities and Exchange Commission reported higher numbers [47]. The report shows an increase in whistleblowing tips to the Commission going from 3000 tips a year to almost 4000 tips received in 2015. From these reported statistics, we can see that a whistleblowing system can receive anywhere between just a couple up to more than a hundred tips a week.

This section will show the result of the earlier estimations from the Storage and Data Transfer calculations in the Design chapter (7.5) using the measured rate f_{low}. The rate puts a lower bound on the number total number of interactions the verifiers can make with the server, meaning that the dummy submissions and dummy replies have to be distributed over that rate. For this it will be assumed that the distribution is uniform. For the four indistinguishable requests that can be made every hour by a journalist, two will be dummy submissions and two will be dummy replies. Client integrity verification is assumed to be built into both the dummy submission and reply protocols.

It is assumed that most whistleblowers visiting the service only perform two types of interactions. They either come to make a new submission, or they come to interact with a reply thread. The distribution between the submission visits and reply thread visits is assumed to be uniform. This means that for the number of whistleblower requessts w, half will be submission interactions and half will be reply thread interactions.

For the calculations made in this section, the following settings are used.

- The rate of indistinguishable dummy requests $f = 4$
- The rate of dummy submission requests $f_{submission} = 2$
- The rate of dummy reply requests $f_{reply} = 2$

	$p = 0.5$	$p = 0.75$	$p = 0.99$
$j = 5$	1680	840	33
$j = 20$	6720	3360	134
$j = 50$	16800	8400	336

Table 8.1. Shows supported number of whistleblowers for one week using f_{low}

	$p = 0.5$	$p = 0.75$	$p = 0.99$
$j = 5$	100800	50400	1980
$j = 20$	403200	201600	8040
$j = 50$	1008000	504000	20160

Table 8.2. Shows supported number of whistleblowers for one week using f_{high}

- The size of a submission $d_{submission} = 5\,\text{MB}$
- The size of a reply thread $d_{reply} = 1\,\text{MB}$,
- The number of whistleblower submission requests $w_{submission} = w/2$
- The number of whistleblower reply requests $w_{reply} = w/2$

The tamper detection probability p is tested with the values 0.5, 0.75 and 0.99. The number of journalists j is tested with the values 5, 20, and 50. The number of whistleblowers is tested with the values 10,75,150 per week.

Client Integrity Verification

In order for the client integrity verification protocol to be secure for a certain number of journalists and whistleblowers, as well as the set tamper detection probability p, recall the equality that must be satisfied or there will not be enough dummies to achieve the wanted tamper detection level. Plugging in for $f_{low} = 4$ we get the following new equality.

$$4j \geq w/(1-p) \tag{8.3}$$

Using this inequality we can also calculate a maximum bounds as the supported number of whistleblowers for client integrity cover traffic based on the probability p and the number of journalists j.

$$w \leq 4j(1-p) \tag{8.4}$$

Table 8.1 shows the number of supported whistleblowers for the client integrity verification for sample values of p and j for one week.

If we instead use the value for $f_{high} = 240$ we get table 8.2 for the supported number of whistleblowers.

Reply Polling Frequency

Equation (7.4) showed the frequency at which a single reply thread would be polled at per hour. Adjusted for the aforementioned settings it would give the following formula, where 2 is the number of dummy requests alloted for reply thread dummies.

$$F = \frac{2}{n_{reply}} \tag{8.5}$$

If the number of reply threads n_{reply} keeps growing, it is clear that polling and cover traffic will become unreliable. For example, any more than 48 active reply threads will mean that any one reply thread is polled less than once per day. This means it will take that long for a journalist to receive any new messages in the thread. Cover traffic will be dependent on this frequency, the number of journalists connected to the thread and how often the whistleblower checks for new messages. The cover traffic will only work to the accepted degree p_{reply} so long as the following inequality is true.

$$\frac{2j}{n_{reply}} \geq \frac{w_{reply}}{1 - p_{reply}} \tag{8.6}$$

where w_{reply} is the frequency per hour at which a whistleblower would visit the reply thread.

As an example, if there are only 100 active reply threads and five journalists in the system with an accepted detection probability of 0.5, the whistleblower will be covered so long as they do not visit more than once every 10 hours. More reply threads and any higher accepted detection probabilities would mean even less frequency for how often the whistleblower could safely visit the thread.

This result indicates that the cover traffic for reply threads is likely too slow to provide safety for a whistleblower other than in small use case scenarios if we use the f_{low} rate. Using the f_{high} rate would gives us the following equation and inequality, where 120 is half of the available dummy requests.

$$F = \frac{120}{n_{reply}} \tag{8.7}$$

$$\frac{120j}{n_{reply}} \geq \frac{w_{reply}}{1 - p_{reply}} \tag{8.8}$$

Using the previous example used for f_{low} with the same settings, we would instead get a more generous rate of 6 dummy visits per hour, meaning that a whistleblower could safely (in relation to the detection probability) visit their reply thread more often.

While using f_{high} seems to make the system viable for normal use, it does use a setting which is perhaps not recommended by the Tor maintainers and might even be harmful to the Tor network. Because of this I do not further use this setting in the next section.

Data Storage

Equation (7.7) is used to estimate the storage. Filling in the values for the settings gives the following equation.

$$s_{total} = 12jw + 10j \tag{8.9}$$

Table 8.3 shows some calculated values of s_{total} for different values of w and j for one week.

Data Transfer

Equation (7.11) is used to estimate the total data transfer. Filling in the values for the settings gives the following equation.

	$w_{week} = 10$	$w_{week} = 75$	$w_{week} = 150$
$j = 5$	9.0 GB	12.9 GB	17.4 GB
$j = 20$	36.0 GB	51.6 GB	69.6 GB
$j = 50$	90.0 GB	129.0 GB	174.0 GB

Table 8.3. Data Storage for one week

	$w_{week} = 10$	$w_{week} = 75$	$w_{week} = 150$
$j = 5$	350.6 Gbit	358.4 Gbit	367.4 Gbit
$j = 20$	5434.6 Gbit	5465.8 Gbit	5501.8 Gbit
$j = 50$	33 746.4 Gbit	33 824.4 Gbit	33 914.4 Gbit

Table 8.4. Data Transfer for one week. Note that the numbers are in bits, effectively $8 * t_{total}$.

$$t_{total} = 3jw + 10j^2 + 2j \tag{8.10}$$

Table 8.4 shows some of the calculated estimates for one week of data transfer for different values of w and j. The results here indicate that the system may require too much data transfer to be useful, especially for larger amounts of journalists.

Chapter 9

Evaluation

This chapter evaluates the new system based on the Requirements (chapter 4) and the Threat Model (chapter 3) established in the earlier part of the thesis.

9.1 Requirements

This section evaluates the system based on the requirements.

Anonymity for whistleblowers

Whistleblowers are anonymous by the use of a Tor hidden service. This means that whistleblowers must connect via Tor or the use of a proxy. This is the upper bound for the anonymity set for the system.

The anonymity for whistleblowers is further improved by the use of dummy submissions which will obfuscate timing and size meta-data. This prevents an observer from as easily being able to determine the whistleblowers identity based on when a submission was made.

Using javascript cryptography with integrity verification can be said to improve anonymity, since it deters the server admin from looking at the contents of the leak. These contents may otherwise lead to the identity of the whistleblower.

The anonymous encryption method used by submissions also improves anonymity for whistleblowers. It makes it so that it is impossible to determine which journalist the whistleblower sent the submission to. This could otherwise be used to clue in on details about a whistleblower. For example, if different journalists handle different areas of investigative journalism. One journalist could be specialized in tax evasion whereas another journalist is specialized in government spying. If an observer sees the tax evasion journalist receive the submission then it may deduct that the leak is about that. Using anonymous encryption makes it impossible for the server admin or network observers to determine who the recipient is.

Integrity

The system features three different types of detection protocols to try to improve integrity. The first of these protocols is the client integrity verification, which is to detect if an adversary in control of the server tries to alter the client script. The other two protocols are dummy protocols for the reply and the submission objects, which try to detect if an adversary in control of the server tries to drop or ignore changes made to replies or submissions.

Use of anonymous encryption should also improve the integrity of the system, since it makes it more difficult for an adversary in control of the server from dropping specific submissions. For example, to a specific journalist. This means an adversary would not be able to target a specific journalist. Since the dummy submissions are kept track of by each journalist, they would also be alerted if their dummies were not present. A malicious server would therefore only have a choice of dropping none of the submissions or at a high chance be detected when attempt to an arbitarily chosen submission.

Encryption of the reply thread is authenticated too, meaning that tampering should be detected.

Confidentiality

Data transmitted into the system is encrypted by one or more journalist keys and should be encrypted before it leaves the whistleblower client by the use of javascript cryptography. Meaning that the system is end-to-end encrypted and the content sent between whistleblowers and journalists is confidential.

Metadata around a submission, such as the recipient, timing and size are to an extent made confidential too. The recipient is confidential by the use of anonymous encryption. The exact timing and size metadata are obfuscated by the use of dummy cover traffic. Reply thread length is obfuscated by a large static padding.

Availability

The use of a Tor hidden service makes it more difficult for authorities to seize the physical server. Although it may also make it less available for less technical users to access the service.

Use of dummies can help with availability as a form of continous testing of the service. This will tell journalists when the service is down, although there may be some ambiguity whether or not this is malice from the server admin's part. The server admin could be purposefully censoring the service for example.

Usability

The system suffers a bit for usability by the use of a Tor hidden service. This may deter less technical whistleblower users from using the system.

Submission encryption is automatically handled by the whistleblower client, meaning that the whistleblower does not have to encrypt by themselves. The journalist client can also decrypt incoming leaks automatically, meaning that journalists will not have to use complicated cryptography programs. Use of a journalist client also means that the journalist will not have to go and retrieve leaks manually from the service.

9.2 Threat Model

This section returns to the threat model and the privacy leaks established earlier to evaluate the system.

IP Leak

The IP leak is countered by the use of Tor, which will anonymize IP addresses for both whistleblowers and journalists connecting to the service. The server will not see the normal IP connecting and will not be able to use that to determine the whistleblowers identity.

DNS, URL and Lookup Leaks

The different types of Lookup Leaks are countered by the use of an anchored URL on the main domain of the newspaper website. This effectively prevents these leaks since no extra DNS request or specific URL request will occur.

Software Usage Anonymity Leak

Whistleblowers are expected to use the Tor Browser when interacting with the system. This limits the anonymity set to those that use that browser. This is worse than e.g. a normal HTTP based website would be since there are more users using the normal Internet than there are Tor users. However, the benefits of Tor are deemed to outweigh this, since Tor counters other more serious types of privacy leaks.

HTTP Confidentiality and Integrity

HTTP Confidentiality and Integrity are provided by default when the whistleblowers and journalists connect to the system's Tor hidden service. The connection occurs under a similar security to TLS.

Third Party Services and Email

Leaks related to third party services are not relevant since no third party service will be used. Non-anonymous emails are not relevant either, since emails are not built into the system.

Server Confidentiality and Integrity Threats

Server confidentiality leaks are to an extent avoided by the use of javascript cryptography. This prevents the server from seeing the submissions in plaintext. This however relies on the integrity of the javascript client which is hosted on the server. To strengthen the integrity of the server different tamper detection protocols are introduced.

Traffic Analysis Leaks

Traffic analysis leaks are to an extent prevented by the use of Tor. However when Tor is used within an organisation, that usage itself may be rare enough to stand out if the organisation is actively monitoring its employees. Journalists are maybe easier to identify on the network since they run a client to check for leaks, but at the same time anonymized within their own group by the anonymous encryption and polling method used.

Who-Had-Access Leaks

There is not much the system can do against Who-Had-Access Leaks, but the anonymous encryption prevents the knowledge of which journalist had access. That knowledge may otherwise be used to glean details about the contents of the whistleblower submission.

Chapter 10

Discussion

This thesis looked at building a new whistleblowing system which introduces some new features to improve security and privacy. Specifically, javascript cryptography to improve whistleblowing applications, the usage of dummy cover traffic to increase privacy of submissions and verify integrity of the JavaScript, and inter-journalist anonymity using key-private encryption.

As shown in the estimations at the end of the Prototype Implementation chapter (chapter 8), the required data transfer for the system quickly grows very large. It is not certain that such large amounts of transfer would be feasible on a Tor hidden service. However, the estimations were calculated with a worst case scenario in mind, with the max amount of supported whistleblower using the service around the clock. In reality, I think there likely would not be such a high load of users and they would not connect during off hours (such as in the middle of the night). In addition, the constantly polling journalist clients could probably be optimised to poll less and fetch all submissions in a compressed format, since there is no low latency requirement for freshness for the reply or submission messages.

The estimations made seem to suggest that the client verification protocol and the dummy submission protocols are viable. However, the reply thread dummies were far too slow and would require too much data transfer in order to be useful. This is likely because for every new submission, there would be a new reply thread that would have to be covered. This would mean that the number of reply threads that have to be covered would keep growing and at the same time the performance necessary to keep up all that covering traffic would also grow. This would quickly get out of hand.

While the reply threads idea does not seem viable, I believe the dummy submissions and especially the client verification can be useful for a whistleblowing system. The client verification is relatively simple to implement and could be useful for other types of Tor hidden services. It provides a way to give some sort of assurance that the javascript served on the website is still honest, which is otherwise not a solved problem when you can not entirely trust the server. This javascript verification could also be useful for other types of outsourced services that do not feature a lot of dynamic content.

If the system is created without the reply thread dummies, the system would be viable for use and likely improve upon the existing solutions. The downside being that the metadata around replies is not protected the same way as initial submissions. On the other hand, replies are still unlinkable to the original submissions, meaning that if there are a lot of submissions and active threads, there will still be a fair amount of anonymity between the whistleblowers and journalists.

The cover traffic protocols rely on the fact that requests can be made indistinguishable. This only works because of Tor, and the main usage of Tor Browser which makes it likely that most

Tor users use the same browser. In comparison, the normal Internet has a variety of browsers, with a large amount of different fingerprints in the forms of headers, installed plugins, etc, that can be used to identify users. Outside of the Tor network, it is unlikely that this method of emulating a browser would work.

In practice, journalists may not be able to run their clients and verification around the clock, leading to scenarios where the whistleblowers become vulnerable after the journalists goes home from work. One alternative here is to enforce that journalists keep their workstation on at all times, or that the verifiers can be run separately on trusted machines with high availability.

This thesis did not look at specific security problems such as web based attacks like cross-site scripting, or weak settings for TLS connections. It should be noted that this does not mean that those problems are not important, and serious security vulnerability will more than likely be usable for the purpose of deanonymization. Instead this thesis looked at features (or the lack thereof) which can be included by design.

In a discussion I had with Swedish newspaper Aftonbladet, they claimed to have around 1000 new submissions every day. However most of these cases were trivial (e.g. company press releases) and the source likely did not require anonymity protection. One could imagine that the system described in this thesis can be used for the fewer sources that do require stronger anonymity or perhaps in cases where the initial contact is already established between a whistleblower and a journalist.

It would have been valuable to get more concrete usage data from journalists. To know exactly how many sources they communicate with, the level of security needed by these sources, the size of submissions etc, but the news agencies I communicated with were not forthcoming with this information. The original plan for this thesis was to work together with a news organisation to produce something that would be deployed in the wild, but it was not possible to find an organisation willing to cooperate in time. The lack of real data does impact the estimations made at the end of the thesis a lot, without knowing how many whistleblowers to expect it is difficult to tell if the system would be useful in reality.

The threat model employed by the thesis may have been a bit too restrictive by implying that the server can not be fully trusted. It is valid for the many cases where news organisations host their service on a hired machine in some other organisations datacenter because that organisation might for example be forced to comply to government orders in secrecy. However, the news organisation could make sure to host the server in a physically trusted location inside their office. Which incidentally is also what SecureDrop recommends adminstrators to do. In any case, working with a different model where the server has more trust the integrity verification protocol would not be as critical. Instead the verification protocol could work as a sort of intrusion detection, and the dummy protocols would still anonymize the time metadata around submissions.

10.1 Comparison with SecureDrop and GlobaLeaks

The system described in this thesis improves upon the existing solutions SecureDrop and GlobaLeaks in several ways. It encrypts whistleblower submissions before they are sent to the service, whereas SecureDrop and GlobaLeaks encrypt the submissions on the server. This means that the plaintext documents are at one point available in memory on the server and thus more at risk.

The system also introduces new features that work to anonymize submission metadata. Timing and size of submissions are obfuscated by the use of dummy submissions. The recipient of a submission is anonymized by the use of anonymous encryption.

Where SecureDrop comes out on top is in the use of an airgapped system, meaning that the journalist key and the plaintext never has to touch a networked computer. This will give better confidentiality as it will be harder for an adversary to access the plaintext.

Unique to the system described in this thesis is also the use of a journalist client which automatically polls for new submissions. This polling itself can lead to journalists being more easily identified for a network observer. This feature is not present in either GlobaLeaks nor SecureDrop.

10.2 Ethical Considerations

Throughout the report the ethical standpoint of why a whistleblowing system is a good thing was never explained. I will attempt to argue in favour of one in this section, as well as why it may be a good thing for the society at large to utilize anonymous whistleblowing systems.

Anonymity itself can be seen as neither good or bad. It can be used for both purposes. Anonymity can be used to protect both victims and perpetrators. For example, a criminal can use it to escape justice, as anonymity would make it impossible for the law to identify the wrongdoer. Often this can be considered when the ethics of Tor and other darknets are discussed. The case of child pornography and trade of illegal drugs and weapons is often brought up. In this case, the anonymity trait given by Tor is definitely protecting criminals, and therefore a bad thing.

On the other hand, anonymity can also protect those who need protection. Victims of extortion can safely report the extorter without fear of retribution. For example, a store owner being extorted for protection by a local gang would without anonymity run the risk of the gangs retribution if they went to the police. Anonymity in some cases removes the bad repercussions that could come from reporting a crime. In this sense it protects the victims. Without these bad repercussion, I think it stands to reason that victims and also whistleblowers are also more likely to come forward to report crimes.

Whistleblowing can be done in many ways. In my system, I assume that there is always a journalist who will eventually receive the information and verify it. The information is not necessarily describing a crime. It could be things that are not illegal per se, but things that are viewed as bad behaviour of an organisation or individual, such as tax evasion. Where such a system can be used for bad is when whistleblowers meaningfully submit false information or information that could be dangerous to the public eye. A political opponent could for example submit false rumors to the press in order to give their rival bad publicity. Dangerous information I imagine could for example be plans to make bombs out of easily obtainable chemicals.

In the end, it is the journalist that decides what should and what should not be published. As such the journalist has the most responsibility to ensure that the system is not misused. The journalist needs to verify that the information is not false and they have to consider whether the effect of publishing the information in a sense of greater good.

10.2.1 Effect on Society

This thesis has looked at improving whistleblowing systems by providing more potential anonymity for whistleblowers. The effect on society is probably small since these types of whistleblowing systems are fairly rare to begin with.

If we however imagine that for example Swedish newspapers start using whistleblowing systems such as the one described in this thesis, more whistleblowers may be encouraged to report. These reports may in turn help a society be more transparent and able to prevent corruption.

10.3 Conclusion

Multiple problems have been shown to exist in today's whistleblowing services served by Swedish newspapers. A lot of the newspapers host whistleblower services on a separate domain, which can be easier snooped upon. Other newspapers use a large amount of third party services. Only two newspapers use clientside cryptography, but they do not encrypt file contents sent. None of the newspapers use a Tor hidden service.

The thesis explored the concept of using javascript cryptography to improve upon existing solutions. It was found that by using cover traffic, tampering of the javascript client could be detected up to a certain probability. Further work on cover traffic also looked at sending dummy submissions and dummy replies. It was found that dummy traffic for submissions could be used to obfuscate timing and size metadata of submissions. Use of anonymous (key-private) encryption methods should also anonymize the recipient of a submissions. Dummy cover traffic for replies did not work as it is too slow and likely generates too much data transfer to be useful.

10.4 Future Work

Further work could be done by looking at non-web based solutions. One idea could be to look to mobile applications, where the programmer has more direct access to lower level network protocols than that which javascript provides. One could for example look at how to implement a whistleblowing system into an existing newspaper application, which would already have a large user set.

Another idea could be to look at further reducing the chance of attempted censorship. One such way could be to use a decentralized service, as opposed to the type of system described in this thesis which is centralized to a web server. One interesting paper in this area looked at sending dummy replies from unknowning website visitors to a backend mixnet [48].

A recent paper investingating the needs of journalists [49] suggests new areas that the computer science community should approach which could be of use to the journalist profession. First contact and authentication, metadata protection and knowledge management are some of the problems that the authors of the paper describe.

Other work could be to look at different methods of javascript integrity. Currently existing solutions are aimed at protecting against third party tamper, with a trusted server model, or they require that the client has the javascript pre-downloaded as a standalone client.

Bibliography

[1] Wikipedia, "Edward snowden." https://en.wikipedia.org/wiki/Edward_Snowden. Accessed: 2016-01-25.

[2] D. L. Poitras, "Citizenfour," 2014. Film/Documentary.

[3] E. F. Foundation, "How the nsa's domestic spying program works." https://www.eff.org/nsa-spying/how-it-works. Accessed: 2015-01-25.

[4] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, 2 1981.

[5] "The tor project." https://www.torproject.org/. Accessed: 2016-01-18.

[6] "I2p anonymous network." https://geti2p.net/en/. Accessed: 2016-01-18.

[7] M. Möser, "Anonymity of bitcoin transactions," *Communications of the ACM*, 9 2013.

[8] "Globaleaks." https://globaleaks.org/. Accessed: 2015-11-17.

[9] "Securedrop." https://securedrop.org/. Accessed: 2015-11-16.

[10] "Wikileaks." https://wikileaks.org/. Accessed: 2015-11-24.

[11] J. Förbundet, "Källskydd." https://www.sjf.se/yrkesfragor/yttrandefrihet/kallskydd. Accessed: 2015-11-02.

[12] D. Nyheter, "SÄpo ska utreda fra läckan." http://www.dn.se/nyheter/politik/sapo-ska-utreda-fra-lackan/. Accessed: 2008-07-01.

[13] PTS, "Uppgifter som ska lagras för brottsbekämpande ändamål en vägledning." http://www.pts.se/upload/Ovrigt/Internet/Tradala/Vagledning_Uppgifter-som-ska-lagras-for-brottsbekampande-andamal.pdf. Accessed: 2015-05-23.

[14] D. Nyheter, "Snabbguide: Vad handlar fra-lagen om?." http://www.dn.se/nyheter/politik/snabbguide-vad-handlar-fra-lagen-om/. Accessed: 2016-03-20.

[15] P. International, "What is the five eyes?." https://www.privacyinternational.org/node/51. Accessed: 2016-01-25.

[16] M. H. Andreas Pfitzmann, "A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," 2010.

[17] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, 1988.

[18] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," *Proceedings of the 2nd international conference on Privacy enhancing technologies*, 2002.

[19] B. M, B. A, D. A, and P. D, "Key-privacy in public-key encryption," *Advances in Cryptology - Asiacrypt*, 2001.

[20] IETF, "Rfc4880 openpgp public-key encrypted session key packet." `https://tools.ietf.org/html/rfc4880#section-5.1`. Accessed: 2016-02-28.

[21] G. N. Michel Abdalla, Mihir Bellare, "Robust encryption," *7th Theory of Cryptography Conference, TCC 2010*, 2010.

[22] T. T. Project, "Tor: The second-generation onion router." `https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf`. Accessed: 2016-01-18.

[23] T. T. Project, "What is the tor browser?." `https://www.torproject.org/projects/torbrowser.html.en`. Accessed: 2016-01-24.

[24] E. F. Foundation, "Https everywhere." `https://www.eff.org/HTTPS-everywhere`. Accessed: 2016-01-25.

[25] "Noscript." `https://noscript.net/`. Accessed: 2016-01-25.

[26] N. Mallesh and M. Wright, "Countering statistical disclosure with receiver- bound cover traffic," *Proc. European Symposium on Research in Computer Security (ESORICS '07)*, 2007.

[27] "Adrian lamo on bradley manning: 'i knew my actions might cost him his life'." `http://www.theguardian.com/world/2013/jan/03/adrian-lamo-bradley-manning-q-and-a`. Accessed: 2013-01-03.

[28] Wired, "Oops! did vice just give away john mcafees location with photo metadata?." `http://www.wired.com/2012/12/oops-did-vice-just-give-away-john-mcafees-location-with-this-photo/`. Accessed: 2012-03-12.

[29] A. C. L. Union, "National security letters." `https://www.aclu.org/national-security-letters`. Accessed: 2016-01-25.

[30] S. R. CNET, "Nsa likely targets anybody who's 'tor-curious'." `http://www.cnet.com/news/nsa-likely-targets-anybody-whos-tor-curious/`. Published: 2014-06-03.

[31] IETF, "Rfc4880 openpgp message format." `https://tools.ietf.org/html/rfc4880`. Published: 2016-01-25.

[32] IETF, "Alexa top sites in sweden." `http://www.alexa.com/topsites/countries;0/SE`. Accessed: 2016-03-20.

[33] "The guardian securedrop." `https://securedrop.theguardian.com/`. Accessed: 2016-01-20.

[34] "The washington post securedrop." `https://www.washingtonpost.com/securedrop/`. Accessed: 2016-01-20.

[35] "Securedrop overview." https://docs.securedrop.org/en/latest/overview.html. Accessed: 2015-11-16.

[36] "Tails - the amnesic incognito live system." https://tails.boum.org/. Accessed: 2016-03-20.

[37] "Grsecurity." https://grsecurity.net/. Accessed: 2016-01-20.

[38] "Ossec: Open source hids security." https://ossec.github.io/. Accessed: 2016-01-20.

[39] "Apparmor." https://wiki.ubuntu.com/AppArmor. Accessed: 2016-01-20.

[40] "The official securedrop directory." https://securedrop.org/directory. Accessed: 2015-11-16.

[41] "Tor2web." https://tor2web.org/. Accessed: 2015-11-24.

[42] "Hermes - about us." http://logioshermes.org/home/about-mission/about-us/. Accessed: 2015-11-17.

[43] "Globaleaks git repository, end2end feature branch." https://github.com/globaleaks/GlobaLeaks/tree/end2end. Accessed: 2015-11-17.

[44] "Dns private exchange (dprive) working group." https://datatracker.ietf.org/wg/dprive/charter/. Accessed: 2016-01-20.

[45] "Selenium - web browser automation." http://www.seleniumhq.org/. Accessed: 2016-03-20.

[46] T. I. Italia, "A voce alta - un anno di segnalazioni," 2015.

[47] U. Securities and E. Commission, "2015 annual report to congress on the dodd-frank whistleblower program," 2015.

[48] H. Corrigan-Gibbs and B. Ford, "Conscript your friends into larger anonymity sets with javascript," 2013.

[49] S. E. McGregor, P. Charters, T. Holliday, and F. Roesner, "Investigating the computer security practices and needs of journalists," *Usenix Security '15*, 2015.