# Secure and Anonymous Communication System

**GSI**
Chia Yuan Cho

**Members**
Daniel Corral
David Powell
Igor Proskurin
Kentoku Matsunami
Michael Pack

## Introduction

In recent years the Internet has seen significant growth. With nearly 400 million people connecting to the Internet, more users have become concerned with anonymity. While the motives for these individuals vary, whether it be protection from government regimes or whistleblowers, their goal is clear: to conceal their identity from would-be watchers. Furthermore, with the increased popularity in social networking many users wish to reconnect with their friends or loved ones with the peace of mind that their conversations remain private. However, popular communication systems such as AIM, Facebook-Chat, or Google-Chat take no measures to provide the end-user security or privacy as they do not aim to provide such measures. Our objective is to achieve a means of secure and anonymous communication; additionally, we wish to obfuscate many of the complexities seen in other software which aims to accomplish identity protection. Our goal is not to revolutionize anonymity tools but to build upon the success of the leaders in this field, such as Tor. Our application was designed with two main objectives: first being security, because we want to eliminate the possibility of passive attacks and for this we make use of RSA encryption in order to transfer our data between the end-users. And second anonymity, such that each user should be guaranteed that their information is not leaked such as who is communicating with who, what the communication consists of, or even when they are communicating or not communicating. In addition, we designed our application to be elegant and simple to use since we believe that the more simple an application is the better it will be on the long run, as such simplicity can be seen throughout software today, especially in the security sector of software, such as the RSA encryption design. With this in mind, we would like to give a brief but detailed look into our program and the steps that were taken along the way to accomplish our goals of security and anonymity.

## Background

Truly private communication on the internet can be very difficult to achieve. Conventional messaging systems often take no precautions to secure their clients' messages, let alone provide them with any sort of anonymity. Given a basic peer to peer (P2P) messaging system that encrypts the contents of messages it is trivial for a network attacker to figure out who the senders and receivers are by simply looking at packet headers. If the messaging system used a central server instead of connecting directly to the recipient and encrypting this connection, a network attacker would have a more difficult time figuring out where the messages were going. This approach however, requires the central server to be a trusted entity. If the central server is willing to cooperate with an attacker then, again we have no privacy.

In more complicated systems like Torchat, packets are encapsulated and sent through an encrypted tunnel via anonymous third parties on the Tor network. Tor creates a degree of anonymity by setting up circuits of peers that act as relays. When a user makes a request on these circuits it is routed through a series of three connected peers before reaching the destination. At each point these peers cannot see the contents of packets and only know about the nodes they are directly connected to. This allows the endpoints to communicate without knowledge of each others location and more importantly keeps network attackers from easily detecting their conversation. The problem with this approach is that Torchat and other similar real time communication systems do not blend into Tor well and are susceptible to timing attacks. Given a large enough view of the network it is possible for an adversary to, over time, connect two communicating users statistically with just packet size and timing analysis.

In addition to traffic analysis, there exists a more sophisticated and powerful attack. This attack does not require a large view of the network and can be used to detect with high confidence the content that a user on the Tor network is accessing. A website fingerprinting attack is a highly effective method of detecting whether an end-user is visiting a particular website or not. Although encryption hides the contents of data and Tor tunnels obscure the endpoints, they fail to conceal the amount of data being transferred. It is possible for a network attacker to create a profile of a user's activities by just using this metric. When a user visits a webpage their browser must download an html page and its associated objects. By creating a vector of these object sizes we can get a pretty good fingerprint of the web page being visited. Once this data has been collected it is possible to compare it against known fingerprints of various websites. These fingerprints are easy to create whether or not the user is connecting to the remote site via SSL because only the objects and their sizes are of concern in this attack. But things start to get more complicated when the user is connecting over a Tor tunnel because individual streams are not as easily distinguishable, but is still very possible using machine learning algorithms. This is actually the most advanced attack on the Tor network today.

Another design that attempts to deal with the problem of anonymity is the mail mixer. In this design all the users of the system send fixed size encrypted messages at predetermined intervals to a central server. The central server then mixes up the messages and broadcasts them to the group as a blob. This keeps observers from being able to figure out which users are speaking to each other. Given a large enough user base this system works quite well for maintaining user privacy. Unfortunately it’s the user base that is itself the problem. These systems have a sort of catch 22 bootstrapping issue. In order for it to be effective it must have a sufficiently large user base. And in order for it to grow its user base it must be an effective system. Additionally, mail mixers do not prevent knowledge of simple participation. In some circumstances it may not be acceptable for an adversary to gain knowledge of a user's participation in such a system. But timing attacks and set-intersection attacks have proved such mail-mixer systems quite vulnerable from anonymity.

**Implementation and Challenges**

The general design of our application relies on several components. We have a specialized client/server component that handles basic message sending/receiving and encryption/decryption. This server/client component interfaces with the Tor network in a few key ways. The server runs as a Tor hidden service within the Tor network. Hidden services were the obvious choice because we do not want our traffic to be readable in the clear at any point as would be the case with an exit node. The client runs as Javascript on top of the Tor browser bundle provided by Tor. Users connect to the server by accessing its Tor hidden services address, a *.onion address, on the Tor network and additionally over SSL.

We chose to put much of the functionality into the client-side so that if the server is compromised there is very little risk to the users. This allows us to have a very minimal server

design that mimics the mail mixer previously described, basically just storing and mixing messages that the clients send. In order to properly mix messages we set a threshold for the minimum number of messages required before displaying. Finally, all messages expire and are deleted from the system after a time period set by the server operator. The client-side has the logic for loading RSA public and private keys as well as local encryption and decryption. We chose to put the cryptographic functionality here so that private keys and plaintext never leave the users system. Additionally, we chose to load the keys from a file rather than copy-paste to prevent clipboard jacking. These two pieces makeup the basic mechanisms for sending and receiving encrypted messages.

The more interesting part of our implementation is how we have chosen to protect against the previously mentioned fingerprinting attack. In order to defend against this attack we crafted our entire system to blend into the noise of 'generic' Tor traffic. Basic http traffic is the most common type of traffic on the Tor network so naturally our architecture is a web server with browser client. Once we had decided on a web app we needed to conduct a field study to find popular destinations on the Tor network in order to further fine tune our fingerprint. With only about a day's worth of data our findings were mostly inconclusive, but we were able to compile a top ten list of sites (see appendix). In order to supplement our small dataset we cross referenced our findings with Alexa.com's top site listing. For each of the candidate sites we considered not only their rank, but also their typical use cases. For example, Facebook and Google were both obvious candidates from the data we gathered, but each would have a distinct fingerprint. Facebook has a much longer interaction time and heavy use of images, whereas Google search is generally very light use with minimal graphics.

Having chosen Google search as our look alike we proceeded to fingerprint it and our application. Rather than attempt to fingerprint each site on Tor we chose to fingerprint in the clear for the purposes of tweaking the app. We followed a similar approach as described in many papers on website fingerprinting and created a vector of object sizes for each site. The results from the Google search fingerprint were very consistent with only a few minor variations in the vectors. Comparing these results to a fingerprint of Facebook or any of the other top sites reveals why this attack is so effective. It's quite easy to distinguish each of the sites using even a rough vector. With the Google search vector results in mind it was a simple, but tedious task to move code around and pad files until our app's vector matched Google's. The result of this tweaking will now cause an analysis of our app fingerprint to yield many false positives for Google.

With all of these components combined we have effectively borrowed from Tor's huge crowd to create an anonymous system that requires no crowd of its own. We can now communicate securely and privately among our group of six without concern of an attacker with a relatively small view of the network becoming aware of our activity.

**Failures and Opportunities for Future Work**

We were unable to conduct long term Tor exit node analysis due to concerns about legal repercussions and so we could not collect highly accurate top visited website information. But with access to a more cooperative network (i.e. not Comcast or Airbears) we were able to collect a decent amount of data (about a day's worth) in order extract top visited website statistics from Tor. Regardless we believe that combining our results with Alexa's global top website list enabled us to choose a reasonable website to imitate.

Our application is not completely protected against attackers with very large views of the network. It is possible for an adversary with a view of both our client and server to, over time, make connections between these two endpoints. This attack does not draw a much concern from our group as the intended use case is time-delayed like the mail mixer rather than real-time like torchat.

It was our initial goal to implement a decentralized communication system that was resilient to government influence. However, after quite a bit of discussion we decided that anonymity was a much more interesting feature and abandoned all serious pursuit. Nevertheless it would be trivial for a user to setup their own instance of the server side code and advertise it to their peers. We feel that this is sufficient for our purposes and allowed us to focus on the more important, core features.

Given more time and experience with the machine learning algorithms discussed in *Website Fingerprinting in Onion Routing Based Anonymizing Networks,* we would have liked to attack our application to prove its resistance. Instead, we followed the findings in the paper to design a fingerprint resistant site. It would also make sense, and be quite trivial, to add an additional layer of "camouflage" traffic as described in the paper. This would further obscure the traffic generated by the site and its users by making concurrent requests to other resources thereby making the detection of object sizes much more difficult. In addition, website fingerprinting is the current "state of the art" passive attack, all timing attacks aside, and as mentioned earlier, we found this attack to be most suitable for our website given the timeline of this project. Furthermore, none of our group members have the necessary experience or tools to perform a decent timing attack on our website. As such, we firmly believed that by focusing on the website fingerprinting attack we minimized the compromise of "security" or anonymity within our program.

## Conclusion

In conclusion, security and anonymity on the Internet is extremely difficult to accomplish. Our objectives for this project was to construct an application that was both secure and anonymous. By Utilizing the already established crowd of the Tor network we were able to hide our entire system, thus achieving anonymity. Furthermore, by using the well documented and widely accepted public key encryption method we were able to securely pass messages, thus achieving a level of security. With these two components we were able to meet our objectives. In addition, we have also shown that our application may be resilient against the modern attack of fingerprint analysis by forming our packets to mimic that of a standard Google search result. The collimation of these factors provides a robust communication system. With the majority of online content gear with the emphasis on usability, security tends to be ignored. It is in our opinion that one does not have sacrifice in one area to gain in another. If designed right, you can achieve the best of both worlds. A secure and elegant application with user friendly features. While our application may not offer complete protection from all online threats, it is a solid start in the right direction and may inspire future web application developers to build on security services such as Tor.

## Appendix

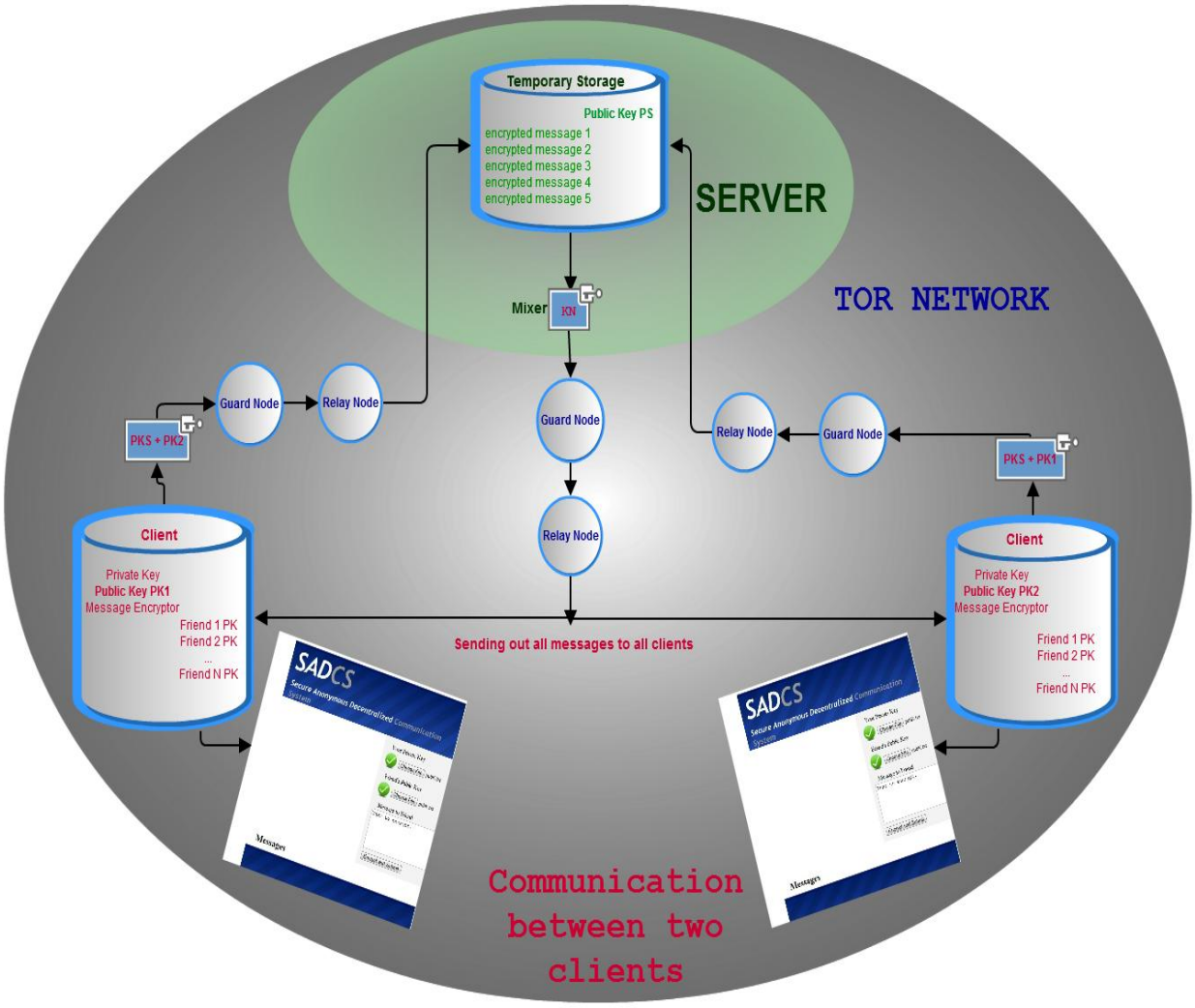


*Figure 1.* High-level view of SACS within the Tor network.

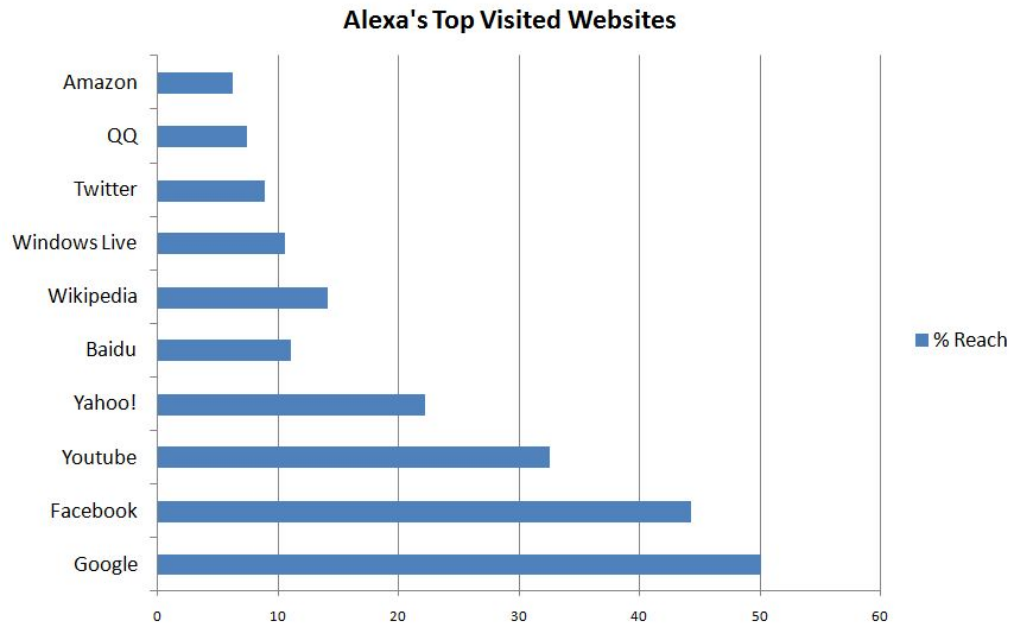


*Figure 2.* Top 10 ranked websites according to *Alexa.com*

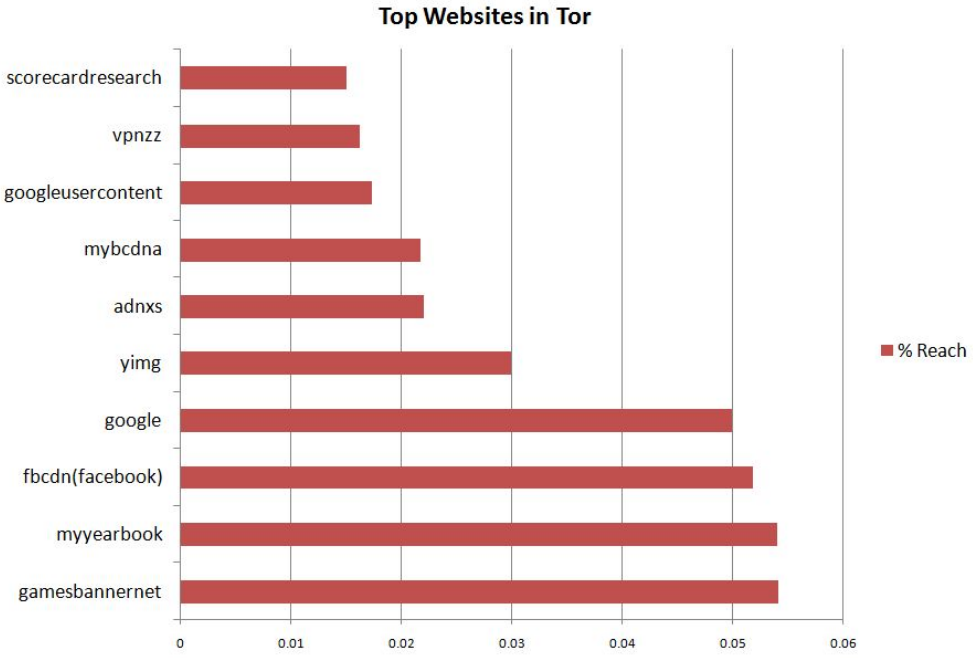


*Figure 3.* Information extracted from a locally run exit node. We found overlapping results in accordance to Alexa's top visited websites.

## Works Cited

"One Cell Is Enough to Break Tor's Anonymity" *"One Cell Is Enough to Break Tor's Anonymity"* Web. <https://blog.torproject.org/blog/one-cell-enough>.

Panchenko, Andriy, Lukas Niessen, and Andreas Zinnen. "Website Fingerprinting in Onion Routing Based Anonymization Networks." *Website Fingerprinting in Onion Routing Based Anonymization Networks*. Web. <http://lorre.uni.lu/~andriy/papers/acmccs-wpes11-fingerprinting.pdf>.

"Preventing Tor DNS Leaks." *Preventing_Tor_DNS_Leaks*. Web. <https://trac.torproject.org/projects/tor/wiki/doc/Preventing_Tor_DNS_Leaks>.

"Tor: Hidden Service Protocol." *Tor: Hidden Service Protocol*. Web. <https://www.torproject.org/docs/hidden-services.html.en>.

"Fingerprinting Websites Using Traffic Analysis." *Drew's Home Page*. N.p., n.d. Web. 10 May 2012. <http://guh.nu/projects/ta/safeweb/safeweb.html>.

"Ronathan Heyning." *Ronathan Heyning*. N.p., n.d. Web. 10 Jan. 2012. <www.eecs.berkeley.edu/%7Edaw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps>.

"pidCrypt - a JavaScript crypto library." *pidCrypt - a JavaScript crypto library*. N.p., n.d. Web. 1 Jan. 2012. <https://www.pidder.com/pidcrypt/>.