USING ENCRYPTION AND TRUSTED THIRD PARTIES TO ENABLE DATA ANONYMITY IN THE FLOCKS ARCHITECTURE: A PRIVACY ENHANCING TECHNOLOGY

Yipeng Liu[1], Martin S Olivier[2]

Information and Computer Security Architectures (ICSA) Research Group

[1]yipeng.liu.sa@gmail.com, Department of Computer Science, University of Pretoria, South Africa
[2]martin@mo.co.za, Department of Computer Science, University of Pretoria, South Africa

ABSTRACT

In recent years, the right of the privacy of the employee and the right of a company to monitor Web traffic through its own network has been heatedly debated. Flocks is a Privacy Enhancing Technology (PET) used within an organization to balance both the need for Web usage anonymity by employees and the need for administrators to trace misusers of the World Wide Web. Flocks operate by establishing a number of Web proxies, which randomly forward requests to each other (or to the destination Web server). User anonymity is preserved as no single proxy can determine the sender of the message.

The current Flocks technology has several shortcomings. One problem is that the communication between the Flock proxies is unencrypted, and thus an administrator of any proxy can read the Web request in plaintext. These logs can compile a huge dossier of information which can be used for illicit purposes against the employee and without the employee's consent.

The aim of the paper is to investigate the use of cryptographic techniques to secure data anonymity while maintaining similar levels of connection anonymity of Crowds and Flocks. We introduce external trusted third parties in addition to the encrypted Flocks system to provide data anonymity for the Flocks architecture. We analyse the new Flocks architecture against various attacks from a threat model. The resulting improved Flocks technology will improve the anonymity of Web users in an organisation.

KEY WORDS

caching proxies, Crowds, Flocks, anonymous browsing, anonymising proxy, privacy enhancing technologies, data anonymity

1 Introduction

During the past few years, Web users have become increasingly aware that their Internet usage is not anonymous. The most commonly used Internet protocols today do not hide the path the data passes through the network, and the packets themselves contain information identifying the endpoints of a communication. The technological and commercial barrier of anonymous communication is further impeded by recent political and economic developments, from increased consumer profiling for many ends and purposes to plans of the US Government for Terrorism Information Awareness (DARPA, 2003).

This lack of privacy of Internet browsing has people trying out services that claim to provide anonymous browsing on the Web[1]. Over the years many technologies have been implemented and tested, using approaches such as encryption of data, pseudo identities and decoy data. Although many of these technologies are effective in providing anonymity to various degrees, most of them are not explicitly designed to be used within an organization and do not take caching and forensics of Web pages into account.

One technology that does cache Web pages is the proxy. Research in proxies has changed from saving bandwidth to hiding the Web requester's identity from an external Web server. Proxies also provide the opportunity to cache and log Web requests and Web pages and thus facilitate an audit trail of a user's Internet activities. The success of proxy in anonymous services can be seen in anonymizer.com (2005), which although simple in architecture, seems to balance user anonymity against a weak threat model to achieve commercial success in Web browsing user anonymity.

Several systems have later been designed to improve on the traditional proxy design. These anonymising proxies have the same goal of achieving sender anonymity, but differ in their design and services offered. We will focus on Crowds (Reiter & Rubin, 1999) and Flocks (Olivier, 2004) and will improve and extend the ideas of the design to achieve better data anonymity.

The current paper will attempt to address the question of how to achieve data anonymity in Flocks. This question is answered by placing a trusted third party between Flocks proxies and the external destination server. We consider how to encrypt the Web pages in all the proxies such that each Flocks proxy administrator cannot read the encrypted cached Web pages but the proxy can provide the user with the required Web pages from their encrypted caches for the user to decrypt. From this discussion, the paper provides a stronger Flocks architecture against attacker threats and provides data anonymity for Web users.

The paper is structured as follows. Section 2 reviews the necessary background about PETs in general and Flocks in particular. Section 3 considers our proposed system of encrypted Web requests and caches and the uses of trusted third parties. We consider limitations and threats to the new architecture in section 4 and conclude our work in section 5.

[1] www.all-nettools.com/library,privacy

2 Overview of Other Systems

Intensive research has been conducted in Privacy Enhancing Technologies resulting in several solutions to the anonymity problem. A fundamental technology for anonymous communication is the proxy. Essentially, a proxy works as a forwarder which accepts requests from computer A and passes it to computer B. In this process, computer A has preserved its anonymity from computer B because computer B thinks that the requests come from the proxy. Examples of proxies include Anonymizer[2] and the Lucent Personalized Web Assistant (LPWA) (Gabber et al 1997; Kristol et al, 1999). Proxy servers are one of the cheapest and easiest ways to deploy within an organization for privacy protection but suffer as a single point of failure because all users need to trust the proxy server. The proxy can centrally log all the requests made by a user and an administrator of the proxy can violate a user's anonymity by browsing and profiling all users' activities.

Other schemes of anonymity are based on Chaum's (1981) Mixnet and the use of encryption between proxies to ensure anonymity. Chaining is a central technique in these schemes and achieves stronger security by sending a message through several anonymous servers and re-encodes the message such that each server only knows the previous server from which the message arrived and the next server to which the message is going. Logging at these servers thus does not reveal the sender or the receiver. A mix is a proxy which accepts messages encrypted with its public key, decrypts them, reorders them randomly, pads them to a constant size and passes them along to their destination, eliminating all evidence of their origin. Over the years, several variations of Mixes have been proposed. In timed mixes (Serjantov and Newman, 2003), messages in a mix are stored in a batch and then flushed at a given time interval. In threshold mixes, messages are stored and then flushed when the batch reaches a certain size. In continuous mixes (Kesdogan et al, 1998), a sender selects a delay from an exponential distribution and adds it to the message sent to the mix. The mix delays the message for the given time period before forwarding it. Besides the various types of mixes, the routing strategy for a message through such mixes is also important. Mix networks (Rennhard and Plattner, 2003) use a free route strategy and mix cascades (Dingledine and Syverson, 2002) restrict the path a message may take through the network. Berthold et al. (2001) argue the case for using mix-cascade in favour of the more common mix-network. By utilizing nested public key cryptography as well as the padding of messages and decoy messages, Mixes generally offer stronger anonymity than Flocks, however Mixes suffers from high timing delays due to its high security levels and often require complex configurations to make it work properly. Flocks achieves better performance because the participant payload in Flocks is entirely independent of the size of the Flock. Thus Flocks has a nice scalability property.

Other implemented anonymizing systems include Onion Routing, Crowds, JAP, Tarzan and Freenet. In onion routing (Reed et al, 1996; Reed et al 1998; Syverson et al, 1997), the sender predetermines the route the message will follow and uses layered public key encryption to send messages across distributed onion routers. Tor (Dingledine et al. 2004), the second generation of onion router addresses limitations in the original design by adding implementation enhancement, integrity checking, perfect forward secrecy and

[2] www.anonymizer.com

other improvements. In Crowds (Reiter and Rubin, 1999), each router uses link-to-link encryption and randomly chooses to either forward the message to another router or to the final destination. JAP (2004) is a mix cascade anonymizing proxy developed in Germany to provide anonymity geared towards low-latency requirements such as Web-browsing. Tarzan (Freedman and Morris, 2002) which is most similar to onion routing but is a peer-to-peer anonymous overlay network that provides generic IP forwarding. Freenet (2004) is an anonymous document publication and distribution service that uses encrypted data storage, geographical distribution and anonymous communication between nodes. All these technologies illustrate approaches to achieve sender anonymity, but unlike Mix Crowds, they are less feasible to use in a local Web proxy in an organization and are difficult to trace back to a sender during an investigation.

3 How a trusted third party (TTP) Flocks works

Our system, TTP Flocks provides similar anonymising Web browsing services to the Crowds (Reiter & Rubin, 1999) and Flocks (Olivier, 2004) concept. There are two broad types of anonymity: data anonymity and connection anonymity. Connection anonymity protects the identity of the user by disguising the communication path between the user and the rest of the world while data anonymity protects the identity of the user by careful modification of the data the user exchanges with the world. Crowds and Flocks are mainly designed to achieve connection anonymity. The aim of TTP Flocks is to retain the advantages offered by Crowds, as well as to introduce data anonymity. We will therefore briefly discuss Crowds and Flocks, as well as how our new architecture can improve on the design.

3.1 Crowds and Flocks

A Crowd consists of participants who want to be anonymous. It is the goal of Crowds to enable Web surfing that is anonymous to various attacks and has acceptable performance in a scalable system. Crowds assumes the premise that if a message is passed around within a network of computers (a Crowd) before being sent to the Web server, an observer can not identify the actual sender. Each member of a crowd operates as an anonymizing HTTP proxy server that can be used by other members of the crowd. When a participant needs to initiate an anonymous connection, it sends its request to another participant. In Crowds, the participant selection strategy is to pick up a proxy randomly from the crowd. On receiving the request, the participant either forwards the request to another proxy or finally submits it to the Web server. This selection of whether to send it to another participant or to the destination Web server is a length control strategy, and is a random decision based on some system-wide parameter $P_f > ½$ (where P_f is the probability of forwarding the request to another proxy). The final destination of the request message (i.e. the Web server) can only conclude that the message came from a member of the Crowd, but cannot tell from which member of crowd the message actually came from. Connection anonymity is thus preserved.

The Crowds system also has link-to-link encryptions and a path key. Each intermediate link in the connection path decrypts the incoming message with the shared keys with its predecessor and encrypts the outgoing messages with the shared keys of its successor. However, since the each intermediate proxy needs to decrypt each message and thus able to view it in plaintext, Crowds does have a path key which is shared with

all proxies in a connection path to enable data anonymity. Each message is thus first encrypted using the path key and then the link-to-link key. Only the last proxy needs to decrypts the message with the path key. The rest of the proxies do not decrypt using the path key in transit. This path key is, however, essentially unneeded; since any intermediate proxy can decrypt the request using the shared path key if absolutely needed. The link-to-link encryption keys is also established by using keys distributed by a "blender", a TTP, which, if compromised can yield all the keys used by all anonymous connections in the Crowd.

The Crowds system is also vulnerable both to the global passive attacker and to corrupted proxy members. A global passive attacker can observe the flow of a message request and trace the originator, while corrupted member proxies can increase the probability that a given member of the Crowd is the originator of a request.

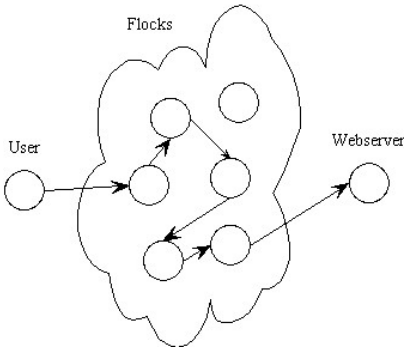
Flocks (Olivier, 2004) examined the most important parameters that influence the effectiveness of anonymity of a Crowds-like system, such as the Crowd size N and the routing parameter P_f. It also considers the use of caching as a means of improving performance and increasing anonymity. Flocks is designed to be used within an organization, and is thus controlled by a central authority. It can be deployed centrally or distributed across many departments. This therefore eliminates some advantages offered by Crowds, since Crowds is a world-wide distributed architecture.

Our solution of TTP Flocks will introduce a secondary authority to balance the central authority of Flocks. We will also introduce data anonymity into the new architecture and thus limiting the auditing ability of the controlling organization.

3.2 TTP Flocks

One limitation of Flocks is that it does not encrypt the Web request that travels through the network, therefore, any observer on the network can easily view the request in plaintext. We thus need to hide the request data moving through the network against eavesdroppers. There are several ways to choose from to encrypt the data. Unlike Chaum's Mixes, we do not want or need total anonymity since we need ways to trace the connection back to the originator of a request during a forensic investigation (Olivier, 2005). We can trace the originator, for instance, when all the proxies work/collude together to trace from any intermediate proxy back to the initiator of the request (Olivier, 2005). Thus the protection against colluding proxies needs to be limited to enable legitimate tracing of a connection.

The Flocks architecture essentially has the same architectural design as Figure 1. To enable easier encryption of Web requests and Web pages and to establish a secondary central authority, we introduce a trusted third party in conjunction with the Flocks proxies. Specifically, we establish a trusted third party between Flocks and the external Web servers. As illustrated in Figure 2, all communications need to be passed from the user to Flocks to the TTP and finally to the Web server.



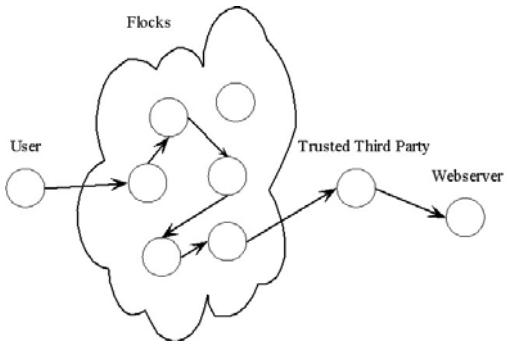


Figure 1: Traditional Crowd/Flocks architecture *Figure 2*: TTP Flocks

With this new architecture, it is easier to enable the encryption of Web request and the caching of encrypted Web pages. We use an example where a user requests and receives a Webpage to illustrate how the anonymous connections are set up. We assume that a user choose to use an anonymous connection from her computer to two anonymity proxies, P_1 and P_2 (Figure 3). The user then makes a HTTP request from her browser to a Web server. All data is therefore sent from the user to P_1, P_2, TTP and finally to the Web server and back.

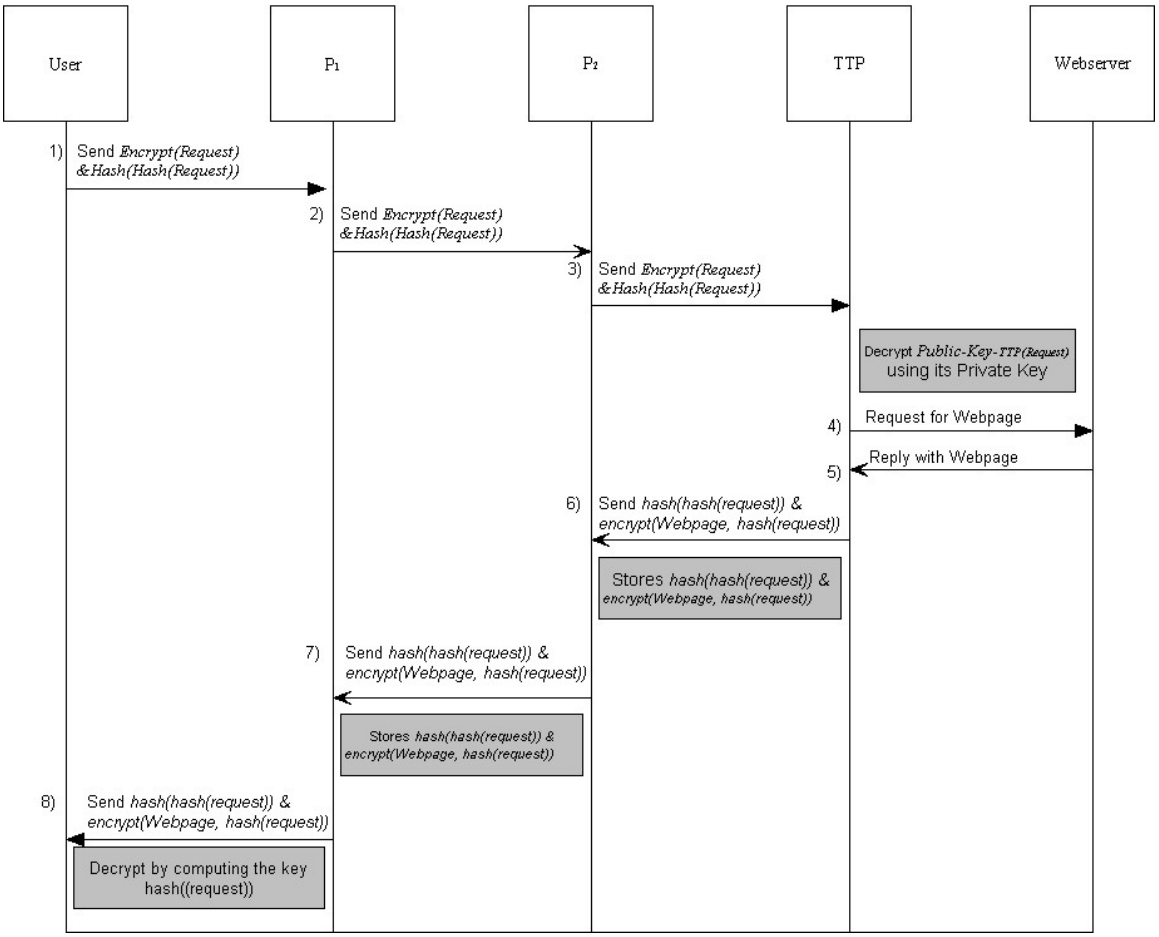


Figure 3: Connection and data flow in TTP Flocks

3.2.1 Sending the request

Since the TTP is the last link of every anonymous connection and has the ability to send Web request and receive Web pages directly from the Web server, the TTP must have the ability to read both the Web request and Webpage in plaintext. The rest of the intermediate proxies, however, do not need to know what is contained in the Web requests or the Web pages. We can therefore hide the content of the Web request from all the Flocks proxies and only reveal it at the end of the connection, i.e. the TTP.

Before the user sends the data to P_1, she encrypts the Web request with the public key of the TTP which is known to all users and proxies. In this way, as the request moves through the Flocks network, it is encrypted and thus unreadable to all Flocks proxies. However, when the data arrives at the TTP, the TTP can decrypt the request using its private key and then send the request in plaintext to the external Web server.

Besides sending the public-key encrypted Web request, the user also needs to compute and send the hash of the hash of the Web request, i.e. double hash the Web request. This is because the doubly hashed Web request is needed to identify the cached copies of encrypted Web pages in each proxy before it is being sent to the TTP (see "Receiving the Webpage" below to understand why this must be done).

The user thus send two pieces of information to the TTP: *Encrypted-using-Public-Key-TTP(Request)* for the TTP and *Hash(Hash(Request))* to match encrypted copies of Web pages in each Flocks proxy. Finally, since the Web request (the URL) usually is a very small amount of text, public key encryption is a quite efficient way to encrypt the data.

3.2.2 Receiving the Webpage

When the external Web server replies back to the TTP with the Webpage, the TTP needs to automatically send out additional requests for images, and other non-text content, or alternately removes contents such as Active-X and JavaScripts. When the TTP finishes collecting all the elements of the Webpage, it will send the Webpage back to the user. To do this, the TTP encrypts the Webpage to prevent Flocks proxies from viewing the Webpage in plaintext. We also need the Flocks proxies to cache both the encrypted Web request and encrypted Webpage for future retrieval (similar to a lookup key and its associated content in a database), thus we need to match the request with its associated Webpage. For this to work, we encrypt the Webpage as follows: Firstly, the TTP encrypts the received Webpage using the hash of the request as the key. Secondly, to encrypt the request to send back to the Flocks proxies, the TTP computes the hash of the hash of the request or alternately, using a different hashing algorithm (than the hashing algorithm which encrypted the Webpage) to compute the hash of the Web request. This is because we do not want the Flocks proxies to be able to decrypt neither the Webpage nor the Web request being sent back. The TTP now sends both the hashed Web request and the encrypted Web pages back through all the intermediate Flocks proxies to the request originator. Thus each intermediate proxy in Flocks stores *hash1(request)* and *encrypt(Webpage, hash2(request))* or alternately *hash(hash(request))* and *encrypt(Webpage, hash(request))* (where *hash1* and *hash2* refer to two different hashing algorithms). A theoretical concern is that since SHA-1 and MD-5 have been theoretically

proven insecure[3], perhaps some other hashing algorithm needs to be used as a hashing algorithm e.g. SHA-256.

When another user wishes to retrieve the same Webpage, each proxy in the connection chain checks whether it is a matching Webpage before it establishes a new connection with another Flocks proxy or to the TTP. The proxy compares the hashed Web request with its cached list of hashed Web request. If there is a match, the proxy then returns the encrypted Webpage. Since, only the user knows the plaintext Web request, she can decrypt the Webpage by hashing the Web request to get the decryption key for the Webpage.

4 Threats and limitations

To properly evaluate the anonymity of TTP Flocks, we need to examine the threats and vulnerabilities of our new system. The security analysis of Crowds has been fairly thoroughly studied (Reiter & Rubin, 1999). From Reiter & Rubin's (1999) analysis, we can derive that TTP Flocks is resistant against local eavesdroppers and end servers if the Flocks size is large. With our TTP, we have also achieved not only connection-anonymity but also data-anonymity, since no participant Flocks proxy can decipher its incoming or outgoing Web request or its encrypted cache. Data-anonymity can be exposed, however, if the TTP is compromised. The attacker will then be able to decrypt all encrypted Web request and Web pages. The TTP is especially vulnerable since it is connected directly to the Internet. However, since data and connection anonymity are designed separately, even when data anonymity is exposed, connection anonymity is still preserved.

An external attacker is an adversary who only has access to data that travels between nodes whereas an internal attacker has controls to the interior workings of a communication node in the system (Wright et al, 2005). We note here that we are now protecting the connection anonymity and not data anonymity. To prevent external attacks, we can encrypt the data differently on each link between each pair of proxies. These encryptions are named link-to-link encryptions. Crowds uses a blender to distribute link-to-link keys; it is centrally located and is fairly permanent, i.e. long-lasting keys. What we need are short-lived encryption/decryption keys that are generated as needed and discarded after use. These keys also needed to have the property that it is impossible to re-derive them from any long-term key material (Borisov et al, 2004). A long term key means that if an attacker intercepts and stores a message, and at a later stage, gain access to the decryption keys by technical or legal means, all messages, past, present and future messages are no longer secure. To provide link-to-link encryption, we use the well-known Diffie-Hellman key agreement protocol (Diffie & Hellman, 1976). Diffie-Hellman allows two users to exchange a secret key over an insecure medium without any prior secrets. However, since Diffie-Hellman key exchange is vulnerable to a man-in-the-middle attack, we also need to use digital signatures and public key certificates to allow the two parties to authenticate themselves to each other. This may leads to a certificate authority (CA) to issue certificate to the TTP Flocks proxies.

This link encryption, is not, however, enough to protect the anonymous connection from internal attackers. Collusion among Flocks proxies can reveal the source of a

[3] http://www.schneier.com/blog/archives/2005/02/sha1_broken.html

connection. For example, if P_1 and P_2 collude in the example in section 3, then after stripping away the link encryption they see the same data. The only way that we aware of to counter this attack, is using nested encryption suggest by Chaum's mixes and Onion Routing. However, in our case, we have to tolerate colluding proxies since it is a built-in weakness to allow the tracing of a connection during an investigation. Although it provides less anonymity, it is usually more efficient as it does not need to use as many public key encryptions, which are very time consuming.

Consideration must also be given to other passive attacks such as observing user traffic patterns and end-to-end timing correlations and Website fingerprinting (Dingledine et al, 2004). The complexity of these attacks are generally only feasible for global observers and may be beyond a limited observer's capabilities.

Other threats include the many forms of active attack. Active attacks such as denial of service can increase TTP Flocks traffic and eventually shut the system down. The best solution is to increase robustness of the network. Hostile Flock proxies can perform a man-in-the-middle attack by discarding incoming requests and creating new requests to direct the user to a malicious website. Some sort of end-to-end integrity checking is needed to prevent this type of attack.

A possible performance bottleneck and vulnerability weakness is the single TTP itself. Since the TTP sits between Flocks and the Internet, it is vulnerable to attacks and can be a network bottleneck for the whole system. The network bottleneck can be ameliorated by introducing a group of TTPs. A group of TTPs between the Flocks and the Internet can spread the load of the network traffic between themselves. With regards to security, if the TTP is vulnerable and its private key compromised, at most data anonymity is revealed. This means that a compromised TTP will reveal to the attacker all plaintext web requests and replies to and from the TTP. However, connection anonymity will still be preserved and the attacker will still not know which user has sent for or received the Webpage.

An alternative to using Flocks with TTP within an organisation is to employ a Mix network within the organisation. Since the organisation can control all Mixes, they can also perform forensic analysis to trace back the user. However, since Mixes uses layered encryption which is very time-consuming, we feel that the performance of a Mix based network will be lower than Flocks with TTPs. However, more performance evaluation needs to be done to support the advantages of Flocks with TTPs and compare it to other technologies.

Finally, more analysis of the caching of Webpages needs to be done to determine how to update the cached encrypted Webpages. Perhaps some kind of time limit needs to be introduced to force the discarding of cached encrypted Webpages after a certain time.

We believe that the above-mentioned threats and vulnerability are inherent in the original Crowds and Flocks architecture, and is not as a result of introducing a TTP. Since we introduced TTP in an attempt to permit data-anonymity, it does not attempt to patch up all the weaknesses in connection-anonymity. Overall, TTP Flocks enables data anonymity while maintaining similar level of connection anonymity and performance of Crowds and Flocks.

5 Conclusion

The aim of this paper was to propose an improved Flocks architecture for Web browsing that offered data-anonymity in addition to connection-anonymity. The Web users in an organization can thus hide both the contents of their communication and to a limited degree their identity in their Web browsing.

We briefly looked at the structure of Crowds and Flocks and discussed some of their disadvantages. A TTP is inserted between it and the external Web server is then proposed to address the problem of data-anonymity.

We then examined some of the drawbacks and advantages of our new architecture, and the impact on the level of anonymity they may have on the system. We note that although TTP Flocks is still susceptible to some attacks, the advantages offered by improved architecture outweigh the disadvantages.

Future research will further extend the concept of a single trusted third party into several trusted third parties to increase performance and anonymity. We can eventually generalize this architecture into a chain of Flocks and establish metrics to measure participant payload, the link between caching and connection length and the relationship between the size of each network, the number of networks and the number of collaborating rogue proxies. These ideas still need to be formalized and analysed in detail.

Finally, since TTP Flocks is presented in only in conceptual form, it would be interesting to compare its implementation performance with other PETs such as Crowds and Flocks.

References

Anonymizer (2005) *Anonymous Browsing Gets Easier*
http://pcworld.about.com/news/Aug072001id57344.htm

Berthold, O. et al (2001) *The disadvantages of free MIX routes and how to overcome them*. International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability. New York: Springer-Verlag. 30-45.

Borisov, N. et al (2004) *Off-the-Record Communication, or, Why Not To Use PGP*. Workshop On Privacy. The Electronic Society archive Proceedings of the 2004 ACM workshop on Privacy in the electronic society. New York:ACM Press. 77-84.

Chaum, D. (February 1981) *Untraceable electronic mail, return addresses, and digital pseudonyms*. Communications of the ACM, 24(2). 84-88.

DARPA. (2003) *Report to Congress regarding the Terrorism Information Awareness program.* http://www.iwar.org.uk/news-archive/tia/darpa-tia-report.htm.

Diffie, W. and Hellman, M. (June 1976) *New Directions in Cryptography*. IEEE Transactions on Information Theory. 644-654.

Dingledine, R. et al (August 2004) *Tor: The Second-Generation Onion Router*. Proceedings of the 13th USENIX Security Symposium. 303-320.

Dingledine R and Syverson P (March 2002) *Reliable MIX Cascade Networks through Reputation*. Proceedings of Financial Cryptography (FC'02). London: Springer-Verlag. 253-268.

Freedman, M.J. and Morris, R. (2002) *Tarzan: A Peer-to-Peer Anonymizing Network Layer*. Proceedings of the 9th ACM conference on Computer and Communications Security. New York:ACM Press. 193-206.

FreeNet. (2004) *The Freenet project homepage*. http://freenet.sourceforge.net.

Gabber, E. et al (1997) *How to make personalized web browsing simple, secure, and anonymous*. Financial Cryptography '97. London: Springer-Verlag. 17-32

JAP. (2004) JAP - anonymity and privacy. http://anon.inf.tu-dresden.de/index_en.html

Kesdogan, D. et al (1998) *Stop-and-go MIXes: Providing probabilistic anonymity in an open system*. Proceedings of Information Hiding Workshop (IH 1998). London: Springer-Verlag. 83-98.

Kristol, D.M. et al (1999) *Design and implementation of the Lucent Personalized Web Assistant (LPWA)*. Bell Labs TR.

Olivier, M.S. (October 2004) *Flocks: Distributed proxies for browsing privacy*. In G. Marsden, P. Kotz´e, and A. Adesina-Ojo, editors, Proceedings of SAICSIT 2004 — fulfilling the promise of ICT, Stellenbosch, South Africa. 79–88.

Olivier, M.S. (February 2005) *Forensics and Privacy-enhancing Technologies - Logging and Collecting Evidence in Flocks*, Accepted for presentation at the First Annual IFIP WG 11.9 International Conference on Digital Forensics, National Center for Forensic Science, Orlando, Florida, USA.

Reed, M.G. et al (December 1996). *Proxies for Anonymous Routing*. Proceedings of the 12th Annual Computer Security Applications Conference, IEEE CS Press, San Diego, CA. 95-104.

Reed, M.G. et al (1998) *Anonymous Connections and Onion Routing*. IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection. 16(4). 482-494.

Reiter, M.K. and Rubin, A.D. (February 1999) *Anonymous web transactions with crowds*. Communications of the ACM. 42(2). 32–48.

Rennhard, M. and Plattner, B. (June 2003) *Practical Anonymity for the Masses with Mix-networks*. Proceedings of the IEEE 8th Intl. Workshop on Enterprise Security (WET ICE 2003). Linz, Austria. 255-262.

Serjantov, A. and Neman, R.E. (May 2003) *On the anonymity of time pool mixes*. Proceedings of the Workshop on Privacy and Anonymity Issues in Networked and Distributed Systems. Athens, Greece. 427-434.

Syverson P.F. et al (1997) *Private Web Browsing*. Journal of Computer Security Special Issue on Web Security. 5(3). 237-248.

Wright, J. (2005) *Designing Anonymity: A Formal Basis for Identity Hiding*. Unpublished PhD dissertation, University of York.