Anonymous Publishing – CS626 - 9/25/09

Kevin Steuer

This document explains the anonymous publishing concept and few current systems.

Key Findings

- There are many types of anonymous systems
- Each system has advantages/disadvantages to them
- Each system needs to be evaluated independently to the goals it's trying to achieve

1.0 Anonymous Publishing

Anonymous Publishing is a concept where individuals have the ability to distribute materials without exposing their identity. Within computer systems, the problem usually is also directly related or combined with anonymous providing and consuming.

1.1 Motivations for anonymous systems

One of the major concerns with developing a true privacy preserving system is the use for illegal purposes. With various p2p systems having legal issues with the authorities it is often hard to justify creating more anonymous tools that can be used maliciously. For example, a particular system could allow individuals to share copyrighted material without the chance of the authorities to prosecute the violator.

There are many valid reasons to pursue anonymity systems. They can be used to avoid being classified as biased if a person is part of a particular group. Whistle blowing, where job security can affect someone's decision to act on what is ethical, can be voiced. It provides a means for an individual to express opinions against the popular view of a topic without looking like an outcast. These problems can all be solved with anonymous systems. Of course these reasons all create other problems which are outside the scope of this paper.

Some groups have been questioning privacy in the internet. The Electronic Privacy Information Center has concerns about government programs like the FBI's Carnivore monitoring system and the EU's "Conventions on Cybercrime" which allows for interception and recording of digital communication [2]. It is easy to see that this is a highly controversial subject.

In some systems while preserving the identity of the publisher, the finger can be pointed at the provider and/or the consumer, to try to cease this type of activity. The original Napster system was shut down because it had a central index for providers that shared the content. The original publisher may or may not have been known but the authorities were able to prosecute the providers where the physical location of the content was stored and shared under the Digital Millennium Copyright Act which prevents the illegal sharing of copyrighted materials. The other subject that can be targeted is the consumers, the individuals downloading the content.

The goals and criteria need to be evaluated on a system to system basis. There is always a tradeoff with complexity vs. efficiency. Some systems may want to concentrate on only confidentiality while others may concentrate more on integrity or reliability.

1.2 Current systems

Most anonymous publishing systems today are decentralized to avoid a single point of failure. There are many systems that try to solve different problems different ways. There are two main categories of anonymous publishing systems. The first category is connection based anonymity where the system will try to hide the identity of the consumer. The second is category attempts to hide the location of the author and/or content.

Systems like The Anonymizer and Proxymate provide connection based anonymity by using a central proxy server to hide the original location of the requester. The proxy in the middle makes the request look like it's coming from the proxy address preventing identification of the requester. When the request comes back the proxy will forward the reply to real requester. This type of system has a single point of failure since it is a centralized approach.

Mixed systems like Onion Routing provide a layer approach where each hop in the path is a layer. Only the nodes directly next to each other along the path know how to read the layer from encryption. As the request travels each node adds an encryption layer. When the reply returns along the same path, a layer is removed. Another type of connection based anonymity, similar to mixed networks, is the concept of crowds. An individual will not be able to be distinguished any further than a crowd. One advantage crowds have over mixed networks is they use symmetric key encryption, resulting in much less overhead during communication. Freenet and Pulious both fall into the connection based category and will be discussed in detail in the following sections.

The system Rewebber acts as both a connection based and author/content based. It acts as a proxy server with a URL like http://www.rewebber.com/surf-encrypted/Ek(U). Only Rewebber knows how to decrypt the encrypted part of the URL (Ek(U)). The Rewebber server decrypts the location of the content, fetches it, and returns to the requestor. Since Rewebber is like a proxy, it also has a single point of failure. Other networks have been proposed as a network of Rewebber networks to make it work like a mixed network and prevent single point of failure [1].

Another system, used more in the past, is the Anderson's Eternity service. When a publisher wants to release content it is posted to an Eternity service for a nominal fee. The Eternity service distributes the content to a random amount of servers that are part of the Eternity network. This is distributed to prevent from a single point of failure and location of the content. Once a document is submitted to the Eternity service it can never be updated or deleted.

2.0 Publius

Publius is a anonymous publishing system that is resistant to censorship. Publius uses the terms of publishers, who post content, servers, who host the content, and retrievers, who download the content. The system provides support for static content like HTML, pdfs, ps files, etc.

2.1 Publius Goals

The Publius system was designed with the following goals:

- Censorship resistant
- Tamper evident
- Source anonymous
- Updatable
- Deniable
- Fault tolerant

- Persistent
- Extensible

2.2 Publius Design

One major difference with Publius is the number of servers is static. All data in the system is encrypted by the publisher. The key the publisher uses for encryption is split using Shamir's secret sharing scheme and distributed among the servers. With Shamir's scheme, the key is split into n pieces and k of those keys is required to recreate the key.

When a file is added the key is randomly chosen by the publisher and encrypts the file. An MD5 hash is computed on the file and the share together for a unique identifier called name used for the folder name on the server. Each server gets the encrypted content and one of the shares and stores it in the name directory. This process then creates a special URL used to retrieve the data and shares. The URL contains three attributes: share (part of key), name (file directory name), and location (server address). The proxy uses encrypted data so any modification of the URL will result in a failure. It is not clear what technique is used but probably some type of hash on the whole string and is appended.

To retrieve the file a requester makes a request to the proxy hiding the identity from the rest of the servers. The URL first checked to if it has been modified, then it is parsed to get the server location, name of the folder, and which key is at each server. The proxy retrieves the data and keys from the servers. The key shares are combined to create the key to decrypt the content. The requester is confident the file is the same as the published file or an MD5 collision has occurred.

Publius offers and option for the publisher to update and delete the content. To delete a file, during publishing the author has an option to create a password and the server stores the hash of that password. This is used to provide authentication for the same user that initially created the file. The type of hash used for password is not clear. To update a file the author provides a password during creation and the new content is added the same way as adding a new file. Afterwards an update URL is placed in the old file directory pointing to the new URL. The old file is then deleted. If the old file is queried the system will return the updated URL. An example of a Publius URL is
http://!anon!/AH2LyMOBWJrDw=GTEaS2GlNNE=NIBsZlvUQP4=sVfdKF7o/kl=EfUTWGQU7LX=OCk7tkhWTUe=GzWiJyio75b=QUiNhQWyUW2=fZAX/MJnq67=y4enf3cLK/0=. !anon! is interpreted by the proxy server to know how to use the URL. Description of how this is implemented is not described in the paper.

One feature Publish provides is to support uploading a full directory. The HTML is updated to reflect the new URL structure. Updates will not break anything because an old URL will always return the new URL to be interpreted.

2.3 Publius Cons

Publius is vulnerable to DOS by continuing to add content till the storage is full. There is a tradeoff with the Shamir's secret sharing scheme while picking a k value. Too large a k can suffer from data censorship by corrupting data. The system has static servers making it somewhat centralized even though there is not a single point of failure. The proxy knows the publisher and requester.

3.0 Freenet

Freenet is a decentralized, distributed storage, and uncensored p2p network. One feature Freenet fixed from Publius is anonymity for the publisher and requestor. Freenet also assumes it is the publishers' responsibility for file encryption.

3.1 Freenet Goals

The Freenets' design goals were:

- privacy for information producers, consumers, and holders
- resistance to information censorship
- high availability and reliability through decentralization
- efficient, scalable, and adaptive storage and routing.

3.2 Freenet Architecture

When a node enters the Freenet network it allocates some storage space for the network to use.

Each content file to be shared has a global unique identifier (GUID) based on SHA-1 secure hashes. When a file is added the GUID is computed. If the file is large multiple GUIDs may be created to split the file into smaller pieces and distribute the file into subsections in the network. Freenet preserves the integrity of a file by creating a subspace key (public and private). This allows creator to update the file using the private key and allows integrity verification with the public key. The creator can post this public key on a website or forum. The documentation does not discuss the feature to allow the creator to explicitly delete the file but is probably not available because the key is not distributed with the file and it seems an attacker could perform a DOS on a file by deleting and violating the goal of being resistant to censorship.

While searching for a file, it is located with the current node searching its' GUID table and queries the nodes with the next best match. This is a bit misleading because the hash isn't really related to the network architecture. If the node reports it cannot find a match the next closest GUID in the table is tried and so on. This is great for anonymity but very bad for performance. A TTL is on the search and decremented at each hop to prevent from infinite loops. This type of search is called steepest-ascent hill-climbing approach. This architecture means there is no lower bound on the content query.

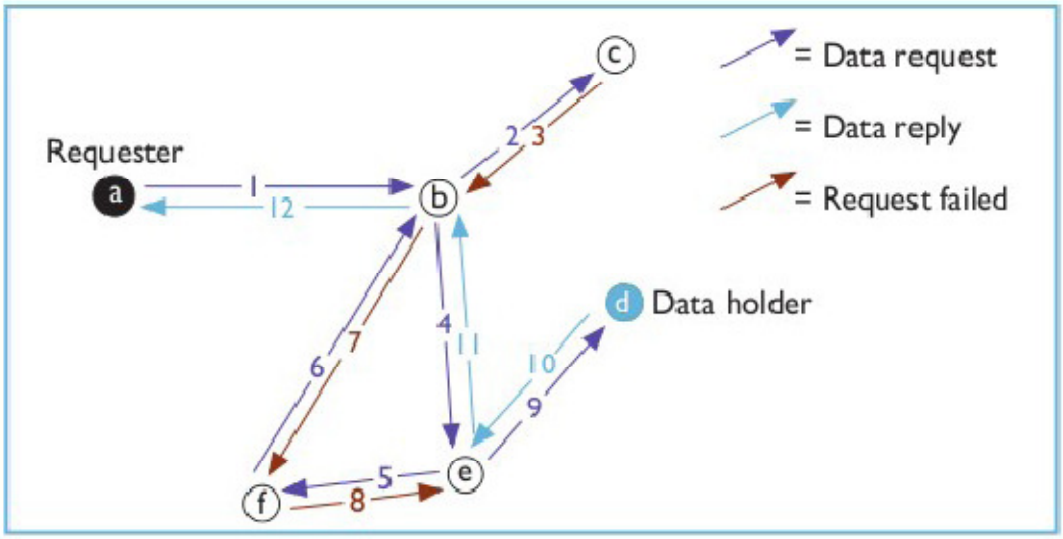

Figure showing a request [2].

Communication within Freenet is limited to each nodes neighbors. A node will never learn of its' neighbors' neighbors. This mixed network scheme allows for connection based anonymity during communication. Again the performance is slower for this tradeoff of preserving the privacy of content storage.

Each node retrains the number of requests for each file. This is used for a priority on the local nodes' files. When space is needed for new files, the lowest priority content is deleted. One problem this raises is over time a file could become less distributed if only a certain part of the network queried a file.

4.0 References

[1] M. Waldman, A. Rubin, and L. Cranor. USENIX Security Symposium 2000. Publius: A robust, tamper-evident, censorship-resistant web publishing system.

[2] I. Clarke, S. Miller, T. Hong, O. Sandberg, and B. Wiley. IEEE Internet Computing Jan/Feb 2002. Protecting Free Expression Online with Freenet.

[3] The Freenet Project. http://freenetproject.org/index.html

[4] Freenet. Wikipedia. http://en.wikipedia.org/wiki/Freenet

[5] Publius Home Page. http://www.cs.nyu.edu/~waldman/publius/publius.html