The Free Haven Project: Distributed Anonymous Storage Service

Roger Dingledine
MIT
arma@mit.edu

Michael J. Freedman
MIT
mfreed@mit.edu

David Molnar
Harvard University
dmolnar@fas.harvard.edu

December 17, 2000

Abstract

We present a design for a system of anonymous storage which resists the attempts of powerful adversaries to find or destroy any stored data. We enumerate distinct notions of anonymity for each party in the system, and suggest a way to classify anonymous systems based on the kinds of anonymity provided. Our design ensures the availability of each document for a publisher-specified lifetime. A reputation system provides server accountability by limiting the damage caused from misbehaving servers. We identify attacks and defenses against anonymous storage services, and close with a list of problems which are currently unsolved.

1 Introduction

Anonymous publication and storage services allow individuals to speak freely without fear of persecution, yet such systems remain poorly understood. Political dissidents must publish in order to reach enough people for their criticisms of a regime to be effective, yet they and their readers require anonymity. Less extreme examples involve cases in which a large and powerful private organization attempts to silence its critics by attacking either the critics themselves or those who make the criticism publically available. Additionally, the recent controversy over Napster and Gnutella has highlighted both a widespread demand for anonymous publication services for non-political purposes, and the consequences of such services failing to provide the anonymity expected.

Systems meeting these needs are just starting to be deployed, and the exact requirements and design choices are not yet clear. Events in 1999 and 2000 have highlighted some shortcomings of already deployed systems; the identification and removal of Napster users who downloaded Metallica songs[30] and the Gnutella Wall of Shame[12] are two examples. These shortcomings led to the development of a new generation of anonymous publication services, such as Freenet[11], which focus specifically on providing anonymity.

It is in this spirit that the Free Haven Project aims to design, implement, and deploy a functioning distributed anonymous *storage* service. We distinguish storage from publication in that storage services focus less on accessibility and more on persistence of data. In the process, we hope to clarify some of the requirements for such systems and highlight design choices.

It is not enough simply to talk about "anonymous" storage and publication. In section 2, we enumerate the many different *kinds* of anonymity which cover different aspects of the system, all important for the realization of a truly anonymous system.

Free Haven meets these requirements with a design based on a community of servers called the *servnet*. Each server, or *servnet node*, holds pieces of some documents. These pieces are called *shares*. In addition,

each servnet node has a persistent identification or *pseudonym* which allows it to be identified by other servnet nodes or potential Free Haven users. Section 3 describes the design of the Free Haven system and the operations that it supports, including inserting and retrieving documents.

We chose to use a network of pseudonymous servers in order to give each server a *reputation*. This reputation allows servers to be 'paid' without needing the robust digital cash scheme required for systems such as Anderson's Eternity Service[2]. Servers form contracts to store given shares for a certain period of time; successfully fulfilling the contract increases the server's reputation and consequently its ability to store some of its own data on other servnet nodes. This gives an incentive for each server to behave well as long as cheating servers can be identified. We show a technique for identifying cheating servers in section 3.7.

The overall idea is similar to the "give up space now, get space forever" scheme used in Intermemory[10], but allows servers to lose reputation if they start behaving badly. In section 3.9 we discuss the *reputation system*, which is the system that keeps track of trust in each server.

Some of the contracts between servers are formed when a user inserts data into the servnet. Most of them, however, will be formed when two servers swap shares by *trading*. Trading allows the servnet to be *dynamic* in the sense that servnet nodes can join and leave easily and without special treatment. To join, a servnet node starts building up a reputation by storing shares for others. To leave, a server trades away all of its shares for short-lived shares, and then waits for them to expire. The benefits and mechanisms of trading are described in section 3.5.

Such a system has powerful adversaries which can launch a range of attacks. We describe some attacks on the Free Haven design in section 4 and show how well the design does (or does not) resist each attack. We then compare our design with other systems aimed at anonymous storage and publication using the kinds of anonymity described in section 6, allowing us to distinguish systems which at first glance look very similar. We conclude with a list of challenges for anonymous publication and storage systems, each of which reflects a limitation in the current Free Haven design.

2 Anonymity for Anonymous Storage

The word "anonymous" can mean many different things. Some systems claim "anonymity" without specifying a precise definition. While the anonymity requirements of communication channels have been considered previously in depth [6, 19], we are not aware of a similar investigation into the requirements for publication and storage systems.

Information is stored in units called *documents*. The *author* of a document is the entity which initially created the document. The *publisher* of a document is the entity which places the document into the system. Documents may have *readers*, which are entities who retrieve the document from the system. An anonymous storage system may have *servers*, which are participants who provide special services required to keep the system running, such as dedicated disk space or bandwidth.

We do not give formal anonymity definitions here. Instead, we attempt to lay the groundwork for future definitions by enumerating different aspects of anonymity relevant to anonymous storage. This enumeration will allow us to compare Free Haven with related work.

In all of these notions of anonymity, there are at least three distinct subnotions based on what the adversary is assumed to already know. A document may be picked first, and then the adversary wishes to learn who authored, read, published, and so on. A user may be picked first, and the adversary wishes to know which documents the user authored, read, published, and so on. Finally, an adversary may know a document *and* a user, and then attempt to confirm its suspicion that the two are linked.

Author Anonymity: A system is author anonymous if an adversary cannot link an author to a document.

Publisher Anonymity: A system is publisher anonymous if it prevents an adversary from linking a publisher to a document.

Reader Anonymity: To say that a system has reader anonymity means that a document cannot be linked with its readers. Reader anonymity protects the privacy of a system's users.

Server Anonymity: Server anonymity means no server can be linked to a document. Here, the adversary always picks the document first. That is, given a document's name or other identifier, an adversary is no closer to knowing which server or servers on the network currently possess this document.

Document Anonymity: Document anonymity means that a server does not know which documents it is storing. Document anonymity is crucial if mere possession of some file is cause for action against the server, because it provides protection to a server operator even after his or her machine has been seized by an adversary. This notion is sometimes also known as 'plausible deniability', but see below under query anonymity.

Passive-server document anonymity means that if the server is allowed to look only at the data that it is storing, it is unable to figure out the contents of the document. This can be achieved via some sort of secret sharing mechanism. That is, multiple servers split up either the document or an encryption key that recreates the document (or both). An alternative approach is to encrypt the document before publishing, using some key which is external to the server.

Active-server document anonymity refers to the situation in which the server is allowed to communicate and compare data with all other servers. Since an active server may act as a reader and do document requests itself, active-server document anonymity seems difficult to achieve without some trusted party that can distinguish server requests from "ordinary" reader requests.

Query-Anonymity: Query anonymity means that the server cannot determine which document it is serving when satisfying a reader's request. For an overview of private information retrieval (PIR), see [31]. A weaker form of query anonymity is *server deniability* – the server knows the identity of the requested document, but no third party can be sure of its identity. Query anonymity can provide another aspect of 'plausible deniability'. This concept is related to deniable encryption[7].

It seems that some of these notions of anonymity may imply each other. We leave this investigation as future work.

2.1 Anonymity and Pseudonymity

So far, we have restricted ourselves to describing anonymity. We extend these notions to allow for the use of *pseudonyms*: if two transactions in the system can be linked, then the attributes which allow them to be linked make up a pseudonym. For example, in an *author-pseudonymous* system, the documents digitally signed by "Publius" could all be verified as "belonging to Publius" without anyone coming to know who "Publius" is in 'real life.'

Both anonymity and pseudonymity protect the privacy of the user's *location* and *true name*. Location refers to the actual physical connection to the system. The term "true name" was introduced by Vinge[48] and popularized by May[33] to refer to the legal identity of an individual. Knowing someone's true name or location allows you to hurt him or her.

Many different actions can be linked to the same pseudonym, while an anonymous system allows no linking at all. This allows the pseudonym to acquire a *reputation*. Free Haven uses pseudonyms to give each server a reputation; the reputation influences how much data a server can store and provides an incentive to act correctly.

2.2 Partial Anonymity

Often an adversary can gain some partial information about the users of a system, such as the fact that they have high-bandwidth connections or all live in California. Preventing an adversary from obtaining *any* such information may be impossible. Instead of asking "is the system anonymous?" the question shifts to "is it anonymous enough?"

We might say that a system is *partially anonymous* if an adversary can only narrow down a search for a user to one of a "set of suspects." If the set is large enough, it is impractical for an adversary to act as if any single suspect were guilty. On the other hand, when the set of suspects is small, mere suspicion may cause an adversary to take action against all of them.

An alternate approach to classifying levels of anonymity is presented by [41], where anonymity levels for users range from "exposed" to "beyond suspicion". These levels are in terms of an idealized adversary's reasonable belief that a user or set of users has performed some particular action.

Independently, Syverson and Stubblebine have developed a logic for talking about the adversary's view of a set of suspects[46]. The logic gives a formal meaning to a "set of suspects" and the notion of an adversary's belief.

2.3 Reasoning about Anonymity

Suppose an author signs his true name to a document before placing it into an anonymous publication system. Is the system still anonymous? This situation raises a crucial question: where does the responsibility of an anonymous publication system begin, and where does it end? What can such a system reasonably be expected to protect? We can give an answer to these questions by explicitly specifying a model for anonymous publication.

We model anonymous publication systems as a single entity (call it Ted) which coordinates communication between other entities in the network. In our model we have a set of senders $\{Alice_i\}$, and a set of recipients $\{Bob_j\}$. When an Alice sends a message to a Bob, Ted receives the message and delivers it to the appropriate Bob. The privacy characteristics of Ted as a communication channel define the level of anonymity that Ted provides.

These privacy characteristics include linkability; ability to reply, persistence of this ability, privacy of this reply; content leaks; channel leaks; persistence of speech; and authorized readers. We emphasize that Ted is not simply a "trusted third party" (despite the name), but provides a specific set of characteristics and does not provide others. For a more complete look at privacy characteristics, look at the first author's thesis [13].

In addition, we will need to complicate this notion with other characteristics, such as reliability of delivery, cost of using a given path, and availability and fragility of the network.

Thus if we can convince ourselves that a given anonymous publishing design is in some sense 'equivalent' to a Ted with certain privacy characteristics, then we can more easily reason about the level of protection provided by that design – by reasoning instead about Ted. In particular, we can ask the question "what is the responsibility of the system" with respect to Ted.

More formally, for each message M_i which $Alice_i$ sends, there is some probability distribution D_i which describes the chance of each Bob being the recipient of the message. If we can replace Ted with a decentralized system which provides an indistinguishable probability distribution for all messages, then we have shown that the decentralized system is equivalent to thi Ted. This may give us an easier way to differentiate between the level of anonymity provided by various projects, because comparing Teds is easier and more intuitive than trying to reason about the effects of trading or caching issues directly.

This description requires significant work before it can become a formal model. For instance, we need to define exactly what we mean by privacy characteristics and enumerate them all; we need to figure out what it means for a probability distribution to be equivalent in this context; and we need to determine exactly how to describe a probability distribution over a complex system like Freenet or Mojo Nation.

3 The Free Haven Design

The overall system consists of the publication system, which is responsible for storing and serving documents, and the communications channel, which is responsible for providing confidential and anonymous communications between parties. This section focuses on the design of the publication system as a back-end for the communications channel.

The agents in our publication system are the **publisher**, the **server**, and the **reader**. These agents are layered over the communications channel; currently they communicate with one another via addresses which are implemented as remailer reply blocks[34]. Remailer reply blocks are a collection of encrypted routing instructions which serve as an address for a pseudonym on the Cypherpunks remailer network.

Publishers are agents that wish to store documents in the service; servers are computers which store data for publishers; and readers are people who retrieve documents from the service.

Free Haven is based on a community of servers called the *servnet*. In this community, each server hosts data from the other servers in exchange for the opportunity to store its own data in the network. The servnet is dynamic: data moves from one server to another every so often, based on each server's trust of the others. Servers transfer data by trading. That is, the only way to introduce a new file into the system is for a server to use (and thus provide) more space on its local system. This new file will migrate to other servers by the process of trading.

Each server has a public key and one (or more) reply blocks, which together can be used to provide secure, authenticated, pseudonymous communication with that server. Every machine in the servnet has a database which contains the public keys and reply blocks of the other servers on the network.

Documents are split into shares and stored on different servers. Publishers assign an expiration date to documents when they are published; servers make a promise to keep their shares of a given document until its expiration date is reached. To encourage honest behavior, some servers check whether other servers "drop" data early, and decrease their trust of such servers. This trust is monitored and updated by use of a *reputation system*. Each server maintains a database containing its perceived reputation of the other servers.

3.1 Publication

When an author (call her Alice) wishes to store a new document in Free Haven, she must first identify a Free Haven server which is willing to store the document for her. Alice might do this by running a server herself. Alternatively, some servers might have public interfaces or have publically available reply blocks and be willing to publish data for others.

To introduce a file F into the servnet, the publishing server first uses Rabin's information dispersal algorithm (IDA) [40] to break the file into shares $f_1, \ldots, f_n$ where any k shares are sufficient to recreate F. The server then generates a key pair (PK_{doc}, SK_{doc}), constructs and signs a data segment for each share f_i, and inserts those segments as new data into its local server space. Attributes in each share include a timestamp, expiration information, the public key which was used to sign it (for integrity verification), information about share numbering, and the signature itself.

The robustness parameter k should be chosen based on some compromise between the importance of the file and the size and available space. A large value of k relative to n makes the file more brittle, because it will be unrecoverable after a few shares are lost. On the other hand, a smaller value of k implies a larger share size, since more data is stored in each share.

We maintain a content-neutral policy towards documents in the Free Haven system. That is, each server agrees to store data for the other servers without regard for the legal or moral issues for that data in any given jurisdiction. For more discussion of the significant moral and legal issues that anonymous systems raise, we refer to the first author's thesis[13].

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.