Trust Economies in the Free Haven Project

by

Brian T. Sniffen

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

Author .....
Department of Electrical Engineering and Computer Science
May 22, 2000

Certified by .....
Ron Rivest
Webster Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Trust Economies in the Free Haven Project

by

Brian T. Sniffen

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2000, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Computer Science and Engineering

Abstract

The Free Haven Project aims to deploy a system for distributed data storage which is robust against attempts by powerful adversaries to find and destroy stored data. Free Haven uses a secure mixnet for communication, and it emphasizes distributed, reliable, and anonymous storage over efficient retrieval. We provide a system for building trust between pseudonymous entities, based entirely on records of observed behavior. Modelling these observed behaviors as an economy allows us to draw heavily on previous economic theory, as well as on existing data havens which base their accountability on financial loss. This trust system provides a means of enforcing accountability without sacrificing anonymity.

Thesis Supervisor: Ron Rivest
Title: Webster Professor of Computer Science and Engineering

Contents

Chapter 1

Introduction

Just build the trust system. It'll be easy

— *Roger Dingledine*

The intent of the Free Haven Project is to create a system for anonymous publication and retrieval of information in such a way that information, once injected into the system, is very difficult to remove. Unlike other anonymous publication systems, it focuses primarily on anonymity, not on availability. Further documentation on the Free Haven Project itself is available at [Din00] and [Din].

1.1 Motivation

> The Internet is moving in the direction of increasing freedom of information and increasingly blurred national boundary lines. At the same time as a strong sense of global community is growing, technical advances have provided greatly increased bandwidth and an enormous amount of computing power and well-connected storage. However, the increases in speed and efficiency have not brought comparable increases in privacy and anonymity on the Internet – indeed, governments and especially corporations are beginning to realize that they can leverage the Internet to provide detailed information about the interests and behaviors of existing or potential customers. Court cases, such as the Church of Scientology's lawsuit against Johan Helsingius[Hel] or the more recent OpenDVD debate[Kro] (and subsequent arrest of DeCSS author Jon Lech Johansen), demonstrate that the Internet currently lacks an adequate infrastructure for truly anonymous publication or distribution of documents or other data. [Din00]

Any attempt to create such an infrastructure will, by nature of its anonymity, require various parties to perform services for people they have never met, with no certainty of their work ever being

repaid. A formalized definition of trust is required. For the purposes of the Free Haven Project, this definition leads us to a *Trust Economy*: a way of trading favors for trust.

An economic model of trust allows us to draw on previous work regarding trust networks with actual financial penalties. It also gives us a tool to talk about net benefits and costs of various security and trust policies.

1.2 Free Haven Project Summary

excerpted from [Din00] The Free Haven Project intends to deploy a system that provides a good infrastructure for anonymous publication. Specifically, this means that the publisher of a given document should not be known; that clients requesting the document should not have to identify themselves to anyone; and that the current location of the document should not be known. Additionally, it would be preferable to limit the number of opportunities where an outsider can show that a given document passed through a given computer. A more thorough examination of our requirements and notions of anonymity can be found in [Din00].

The overall design is based on a community of servers (which as a whole is termed the 'servnet') where each server hosts data from the other servers in exchange for the opportunity to store data of its own in the servnet. When an author wishes to publish a document, she breaks the document into shares, where a subset (any k of n) is sufficient to reconstruct the document, and then for each share, negotiates for some server to publish that share on the servnet. The servers then trade shares around behind the scenes. When a reader wishes to retrieve a document from the servnet, she requests it from any server, including a location and key which can be used to deliver the document in a private manner. This server broadcasts the request to all other servers, and those which are holding shares for that document encrypt them and deliver them to the reader's location. Also behind the scenes, the shares employ what is essentially the 'buddy system' to maintain some accountability: servers which drop shares or are otherwise unreliable get noticed after a while, and are trusted less. A trust module on each server maintains a database on the behavior of each other server, based on past direct experience and also what other servers have said. For communication both between servers

and between the servnet and readers, we rely on an existing mixnet infrastructure to provide an anonymous channel.

The system is designed to store data without concern for its popularity or controversial nature. Possible uses include storing source code or binaries for software which is currently under legal debate, such as the recent DeCSS controversy or other software with patent issues; publishing political speech in an anonymous fashion for people afraid that tying their speech to their public persona will damage their reputation; or even storing more normal-looking data like a set of public records from Kosovo.

Free Haven is designed more for anonymity and persistence of documents than for frequent querying — we expect that in many cases, interesting material will be retrieved from the system and published in a more available fashion (such as normal web pages) in a jurisdiction where such publishing is more reasonable. Then the document in the servnet would only need to be accessed if the other sources were shut down.

The potential adversaries are many and diverse: governments, corporations, and individuals all have reason to oppose the system. There will be social attacks from citizens and countries trying to undermine the trust in the security of the system, as well as attacking the motivation for servnet node operators to continue running nodes. There will be political attacks, using the influence of a country's leaders to discourage use of the servnet. There will be government and legal attacks, where authorities attempt to shut down servnet nodes or arrest operators. Indeed, in many cases ordinary citizens can recruit the power of the government through lawsuits or subpoenas. Multinational corporations will hold sway over several countries, influencing them to pass similar laws against anonymous networks. There will be technical attacks, both from individuals and from corporations and national intelligence agencies, targeted either at the system as a whole or at particular documents or node operators, to reduce the quality of service or gain control of part of the network. Clearly the system needs to be designed with stability, security, and longevity in mind.

Chapter 2

Related Works

2.1 PGP Key Servers

Pretty Good Privacy is a general-use public-key cryptography tool. It provides for encrypted and signed communication. Users exchange their public keys by means of widely-publicized servers:

> Public Key Servers exist for the purpose of making your public key available in a common database where everybody can have access to it for the purpose of encrypting messages to you. While a number of key servers exist, it is only necessary to send your key to one of them. The key server will take care of the job of sending your key to all other known servers.[PGP]

Each public key on the key servers is signed by people who can verify, by some means, that the person whose name is attached to a key actually controls the associated secret key.

Users who download a public key from the servers set two parameters within their own installation of PGP:

- Confidence that this key represents the user whose name is attached.
- Confidence that this person exercises good judgment in signing other people's keys.

By means of these two values, a network of trust is established. PGP serves as an effective means of communication, and has established a good infrastructure for safely exchanging keys. It fails due to user interface problems: most notably, each key to be accepted as an introducer requires informed attention on the part of the user. As a result, it has not become widespread enough to be generally useful.

2.2 Netscape Certificate Authorities

Many commercial web sites wish to ensure that visitors can communicate with them securely. Some also want to strongly verify the identities of their visitors. Certificate Authorities (CAs) exist in a multi-rooted hierarchy. A handful of top-level authorities certify most commercial sites; such sites are then able to establish session keys with their users. Some such sites issue personal certificates to their users.

The flaw here is that users are locked into trusting the established CAs. A user can't decide that he trusts his friends to certify things, but not VeriSign.

2.3 AOL Instant Messenger

AIM[AOL] is a popular messaging client. In order to avoid harassment, users are allowed to file a complaint about those who have sent them messages. Users who accumulate a certain number of complaints per unit time are automatically disconnected from the service. The problem here is that *every* AIM user is therefore trusting *every* other AIM user to act as a censor. AIM has had numerous problems with this complaint feature being used to deny service to various targets.

2.4 The Eternity Service

Ross Anderson's Eternity Service[And] is the motivation for this project. It includes a wonderful vision of how the world might work in the future, in terms of data havens. On the other hand, it relies on a stable digital cash scheme, which is certainly not available today. Furthermore, it has a

stronger correlation between ability to store data into the system and amount of real-world capital available. While our proposal does have a loose correlation between available resources and amount of influence over the servnet, it is not nearly so direct.

2.5 Napster

The Napster service[Nap] is a company based around connecting people who are offering MP3 files to people who want to download them. While they provide no real anonymity and disclaim all legal liability, a very important thing to note about the Napster service is that it is highly successful. Thousands of people use Napster daily to exchange music; if there were greater security (and comparable ease of use), I suspect that many thousands more would participate. Napster presents a very clear argument that the Internet community wants a service like Free Haven.

The response to the Metallica suit[Lew] — in particular, the continued use of Napster for trading of illegal MP3 files — indicates that the Internet community is willing to continue use of this service even knowing that it could have negative legal consequences.

2.6 Gnutella

Gnutella[ea] is a peer-to-peer Napster clone. It was developed by an employee of AOL, then shut down by that company. Development is proceeding in the hands of open-source contributors. Gnutella depends on the "Small Worlds" model to maintain a connected network.

2.6.1 Small Worlds Model

Social networks display two characteristics which initially may appear to be contradictory. First, social connections display clustering, whereby friends are likely to share the same group of friends. Second, they exhibit what has been termed by Stanley Milgram as the "small worlds effect" [Mil67]. Namely, any two people can establish contact by going through only a short chain of intermediate acquaintances. Milgram proposed that all people in the world are separated by six intermediaries

on average; this effect is better known as "Six Degrees of Separation" or the "Kevin Bacon Game."

2.6.2 Flaws

According to the new developers' web site [ea]:

> Gnutella puts a stop to all those shenanigans. When you send a query to the GnutellaNet, there is not much in it that can link that query to you. I'm not saying it's totally impossible to figure out who's searching for what, but it's pretty unlikely, and each time your query is passed, the possibility of discovering who originated that query is reduced exponentially. More on that in the next section.
>
> In short, there is no safer way to search without being watched.
>
> A big however, however. To speed things up, downloads are not anonymous. Well, we have to make compromises. But again, nobody's keeping logs, and nobody's trying to profile you.

They're having some problems maintaining anonymity, however. While Gnutella suggests trusting all of your "friends" to not log your queries, sites such as the Gnutella Wall of Shame[tC], which attempts to entrap child pornographers using the Gnutella service, suggest otherwise. The direct file-transfer portion of the Gnutella service has been demonstrated to not adequately protect the anonymity of servers or readers. The failure is reducible to one of trust: the Gnutella service requires users to trust their neighbors, but provides no means of ensuring that trust is kept.

2.7 Freenet

Freenet is a project akin to Free Haven, but focused on availability, not anonymity. If even a few nodes in the system are corrupted, the entire service becomes untrustworthy for purposes of anonymity. However, Freenet's tendency to widely duplicate popular information makes it difficult to remove data.

The near-total lack of anonymity in Freenet raises the possibility of traditional search-and-seizure attacks, and calls their reliability into question.

2.8 Zero Knowledge Systems

Users of Zero Knowledge Systems'[zks] "Freedom Network" are currently forced to trust ZKS Inc. ZKS sells pseudonyms; right now, it uses the standard credit card system. If subjected to a warrant, ZKS would be forced to reveal the owner of a given pseudonym. ZKS claims plans to fix this problem by switching to an anonymous e-cash system. From a trust-system point of view, this transfers the problem to trusting the maintainers of the e-cash system.

It also raises the point that there currently is no widely available e-cash infrastructure.

2.9 Internet Relay Chat

The IRC network establishes trust based on a very simple model: IP addresses. A common technique among crackers on IRC is to flood the host of a victim, temporarily knocking him off the network. While the victim is thus distracted, the cracker spoofs packets from the victim to an IRC server, giving privileges to himself and removing them from the victim. When the victim returns to the network, he is now unprivileged, and is subject to further attacks by the now-privileged cracker.

Clearly, non-cryptographic trust models are not useful against modern adversaries.

2.10 Mobile Agents for Network Trust

MANET[KBD98] is a DARPA project to produce "a compromise-tolerant structure for information gathering." The motivation is to create a system whereby untrusted networks can cooperate to fight against "mobile adversaries": adversaries who move from one network to another. The MANET project attempts to avoid the problem of corrupted servers by requiring several servers to weigh in on a subject with direct evidence before action is taken. This approach is, even in the eyes of the MANET authors, somewhat naïve. MANET relies on correct execution of mobile code

2.11 Publius

The Publius system[WRC] has an implicit trust model entirely divorced from reality: there is a static set of servers, all of which are widely and publicly known. Documents are staticly stored on several of these servers, having been split using Shamir's Secret Sharing Algorithm. If one of these servers is corrupted, it is assumed it will be replaced.

It's possible to turn Publius into an informal but workable system with a very small amount of work: if the servers are all publicly known, then an informal trust network can be put into place. Create a discussion forum, called perhaps `alt.anonymous.publius`. Sites which wish to become servers advertise here; if users discover that a server has been corrupted, they can denounce it in that forum. Of course, this system provides no formality or assurances whatsoever.

Chapter 3

Trust Networks

This system is built off some amount of trust in the other nodes of the servnet. In particular, the protocol supports some enforcement of good practice by the other nodes, but there is still plenty of opportunity for nodes to obey the rules until they are sufficiently trusted, and then start breaking the rules in subtle ways. For instance, a node might sometimes fail to provide a receipt to a share's buddy during a trade, introducing confusion as to whether the other side is trying to trick the buddy into believing that the share had been traded away. More destructive would be for a node to never query the buddy shares for any of the shares it possesses, or even wrongly accuse a different node of losing that share. The list goes on and on. However, with careful trust management, each node ought to be able to keep track of which nodes it trusts; with the cushioning provided by Rabin's information dispersal algorithm, only a significant fraction of the nodes turning evil at once will result in actual loss of files.

3.1 Protocols for informing neighbors

Each node needs to keep two values describing each other node it knows about: trust and metatrust. The first, trust, signifies a belief that the node in question will obey the dictates of the Free Haven Protocol. The second, metatrust, signifies a belief that the utterances of that node are valuable information. For each of these two values, each node also needs to maintain a confidence rating.

This serves to represent the "stiffness" of the trust value. For example, a node which has Trust 100 but Confidence 1 will be trusted with a great deal of data, but will lose or gain trust rapidly in response to the utterances of other nodes. Exactly how these values are used is left entirely up to each node.

Some nodes may wish to set all meta-trusts to zero, thus ignoring all outside utterances. Other nodes may wish to set confidence values very high, ignoring not only outside utterances but also direct evidence of wrongdoing.

Nodes should broadcast referrals in several circumstances:

- When they log the honest completion of a trade.
- When they fail to verify that a buddy[1] of a share they hold is safely held.
- When the trust or metatrust in a node otherwise changes substantially.

3.2 Getting information

In addition to referrals, the trust system should gain information from the operation of its own node. Such information must be treated as having a very high metatrust — as we're talking to ourselves, why would we lie?

The trust system should log all transactions between its own node and others. It needs this information to determine both when shares have successfully expired and when shares have been deleted before their expiration date.

3.3 Introducing new nodes

One of the most important parts of the design of Free Haven is the capacity to seamlessly integrate new servers. These servers need to be able to join and participate in the servnet simply by installing some simple packages and making contact with public servnet nodes. Such nodes (e.g.

[1] See the Free Haven protocol design.

`freehaven.org` itself) have the option to configure themselves as *introducers*. While nodes configured with the default settings should ignore communications from unknown nodes, introducers respond to such communications by querying the new node for its public key and return address[2].

Provided with this data, the introducer adds the new node to its database, then broadcasts a referral of that node. Other nodes are welcome to do what they like with this information. It is suggested that the initial trust and metatrust values both be zero, with some small amount of confidence in each. The introducer may also wish to send to the new node a referral for each node it knows about, thus assisting in keeping the network as fully connected as possible.

From this point, it is expected that existing nodes will attempt to offer trades to the new node, and vice versa.

3.4 Purging old nodes

Old nodes need to be removed from use after a while, since continued mail bombardment from the mixnet to a closed account will eventually slow down the service (as well as anger a lot of systems administrators). On the other hand, any operation which can remove nodes from the servnet must be handled very carefully, because it's easy to introduce security vulnerabilities that let an adversary disable a node.

One solution is to allow the trust network to gradually lose trust in a given node to the point that it doesn't ever send trade requests to it. On the other hand, broadcast requests will still go to it. This means that we might consider a sort of 'ping' query for a node, to make sure that it's still responding to its mail. On the other hand, this introduces new denial of service attacks where a node might answer pings but not other queries.

Therefore, we've chosen to add a new behavior to the trust system. As the trust database logs all transactions, it is in a position to notice when a multistage transaction fails to complete. In cases where such a transaction fails to complete, the trust system should notice and mark that host as "potentially dead." After a potentially dead host continues to ignore queries for some time, it

[2]Presumably this is some sort of anonymous communications channel, such as a Mixnet.

should be marked "presumably dead." A host should be removed from the "dead" lists as soon as it reopens communications.

3.5 When to trade

While individual node administrators should be free to change this as they wish, the default Free Haven installation should base its unit of trust currency on the product of share size and storage duration: megabyte-months. Each host should also grant a small amount of leeway. This allows new nodes, for example, to be able to trade small shares despite their 0-trust rating.

A reasonable policy will balance trust-increasing trades with low-confidence nodes against guaranteed-safe trades with high-trust, high-confidence nodes.

Chapter 4

Implementation

The Free Haven Trust Module is a library of code accessed by the Haven Module. Because it acts as the information repository for the Haven module, it makes sense to offload its logging and querying needs to a relational database backend. Such a backend can be easily shared with the Communications Module, ensuring coordination of keys and reply blocks. It also makes sense to store metadata about shares in the database, while leaving the shares themselves in a traditional file-system. Appropriate tables for the trust portion can be described in SQL as shown in Figure 4-1.

Of particular note, the `trades.receipt` data is duplicated in the other fields. The receipt itself needs to be there for rebroadcast in case of betrayal, and for its signature. The other fields have been extracted from it for quick and easy access.

Based on user configuration and the above-mentioned database, the trust module supports the following API:

`inform_trustdb(struct tag_t *tag_list)` Takes in a parsed XML referral and adds it to the `referrals` database. See the "Interpreting Referrals" section below for more information.

`trust_find_trade_host(char *target_host, char *desc)` This call gives the trust db great freedom of action: the result should be to find a host we wish to trade with, then write its key into `target_host` and a description of a share it might want into `desc`. This last is obtained

```
create table nodes (
    id              integer unique,
    key             varchar(4000),
    replyblock      varchar(8000),
    trust           integer,
    confidence      integer,
    metatrust       integer,
    metaconfidence  integer
)

create table trades (
    source_node     integer references nodes,
    dest_node       integer references nodes,
    receipt         varchar(4000),
    given_keyhash   varchar(512) primary key,
    given_size_k    integer,
    given_exp       date,
    taken_keyhash   varchar(512) primary key,
    taken_size_k    integer,
    taken_exp       date,
    timestamp       date,
    expired         integer
)

create table referrals (
    target_id       integer references nodes,
    referrer_id     integer references nodes,
    trust           integer,
    confidence      integer,
    metatrust       integer,
    metaconfidence  integer
)
```

Figure 4-1: SQL Table Descriptions

from the sharedb once we have selected a host. In the initial implementation, the choice of host is random, but weighted towards those with low confidence.

`trust_find_trade_host_by_share(char *target_host, const char *share)` Given a share description `share`, we find all the hosts which we'd trust to hold it, then select randomly among them, weighted towards those with high confidence.

`trust_find_desc_for_host(char *desc, char *target_host, char *share)` This call writes a share description into `desc` which we are willing to receive from `target_host`. In the initial implementation, this simply describes the limits of our trust of them.

`trust_accept_share_phase_one(struct tag_t *tag_list)` In the initial implementation, this call checks on the trust of the offering node, to see if we're likely to be comfortable trading an equivalent share to them.

`initialize_trust_module(), close_trustdb()` These are simple initializers which deal with opening a connection to the database and reading in the configuration file.

The two functions `trust_find_trade_host` and `trust_find_trade_host_by_share` are biased in different directions with respect to confidence. The assumption is that the haven module will call `trust_find_trade_host` when it wishes to do trust-building trades and serve the priorities of the trust system, and call `trust_find_trade_host_by_share` when it wishes to do safe trades. In accordance with this assumption, we trade with low-confidence hosts when offered the opportunity to do so, and high-confidence hosts when told it's important.

4.1 Interpreting Referrals

When the Trust Module receives a referral of the form [Trust: T, Confidence: C, Metatrust: M, Metaconf: F] targeted at a node with characteristics [Trust: t, Confidence: c, Metatrust: m, Metaconf: f], it first checks to make sure it has no other referrals with the same target and referrer — if it does, those are backed out of the system and replaced with the new referral.

Then we add $((T - t) \times M \div c)$ to t, and move c one notch in the same direction — that is, if we have increased a positive t or decreased a negative t, we increment c, otherwise we decrement c. Then we add $((M - m) \times M \div f)$ to m, and move f one notch in the same direction, as above.

If we do currently agree with the referral ($T = t, M = m$), we do not change our trust in the target node at all, but instead increment our metatrust and metaconfidence in the referrer: if that node agrees with us, it must be worth listening to.

4.1.1 Referrals with receipts

In the case of a referral with a receipt, we proceed somewhat differently. We ignore metatrust issues, and instead proceed as if we had noticed the lapse ourselves: we decrease our trust in the node which defaulted by the product of the size and the intended duration of the trade (that is, the difference between the timestamp on the trade and the expiration date of the share) and increment our confidence in that trust.

We take no action with regard to the metatrust of the referrer; it was acting based on obvious information, so we have no reason to believe it is particularly wise in other matters..

4.2 Gaining Trust Independently

The trust module periodically scans the database, rebuilding trust information. It is in this way that expired shares are noticed. When a share expires without any sign of having disappeared early, we increase our trust in the node we traded it to by the product of the size and the duration of the trade (that is, the difference between the timestamp on the trade and the expiration date of the share), increment our confidence in that trust, and mark that trade receipt as expired.

4.3 Losing Trust Independently

When the Haven Module fails to verify that a buddy share still exists, it informs the Trust Module. The respone is the reverse of a successful trade, above: we decrease our trust in the node which

defaulted by the product of the size and the intended duration of the trade (that is, the difference between the timestamp on the trade and the expiration date of the share), increment our confidence in that trust, and "squawk."

This "squawk" takes the form of a broadcast referral about this new trust data, including a receipt for the trade which landed that share at the apparantly corrupt host.

4.3.1 Disagreement

When scanning the database as mentioned above, the trust module notices when it has referrals for a node which are of the opposite sign from its current trust. Such referrers have their metatrust decreased, to indicate that they are either unwise or untrustworthy.

Chapter 5

Attacks on the Trust System

There are a variety of attacks which are possible on the Free Haven system. Many of these attacks are far outside the scope of the Trust Module: social attacks on system security and servnet node operators, political attacks to discourage servnet use, government and legal attacks to shut down nodes or arrest operators, denial of service attacks on the communications anonymous channel, and attacks on the infrastructure of the server network.

Some of these attacks, such as temporary denials of service, have negative repercussions on the trust of a node. These repercussions might be qualified as "unfair," but are best considered in the following light: if a node is vulnerable to these attacks, it is not capable of meeting the specifications of the Free Haven protocol. Such a node is not worthy of trust to meet those specifications. The trust system does not judge intent, merely actions.

5.1 Simple Betrayal

The simplest attack is this: Become part of the Servnet, earn trust, then betray it by deleting files before their expiration dates. The trust economy is designed to make this as unprofitable as possible. The size-time currency means that a corrupt node has to donate at least as much to the Free Haven as it removes. This 50% useful work ratio is a rather loose lower bound — it requires duping a great number of high-metatrust nodes into recommending you.

A node which engages in this behavior should be caught by the buddy system when it deletes each share.

5.2 Buddy Coopting

It's possible for a corrupt node to gain control of both a share and its buddy; at this point it can delete one of them without repercussions. This means that corrupt nodes can defeat the buddy system by capturing both buddies, then deleting them.

A possible work-around to this attack involved separating the contact addresses for trading and for buddy checking, preventing corrupt nodes from acquiring the buddies of the shares they already have. Such an approach adds a great deal of complexity, and opens other attack avenues.

5.3 Trading Receipt Games

The receipts used in trading are a complicated mechanism, and we have no formal system for talking about how they interact. While we believe that the signed timestamp makes it clear who did what and when, it's possible that some attacks exit, likely involving multi-node adversaries engaging in coordinated bait-and-switch games with target nodes.

5.4 Pollution

An adversary can join the server network, then trade away garbage for valuable data. A sufficiently wealthy adversary could even purchase a series of very large drives, then trade away enough garbage to have the majority of the data in the server network on his drives, subject to deletion.

We have no defense against this attack. However, any adversary capable of perpetrating the above attack against a widely-used Free Haven is equally capable of many cheaper, easier, non-technical attacks.

5.5 False Referrals

An adversary can broadcast false referrals, or direct them to specific hosts. The metaconfidence system combined with the single-reporting policy provide somewhat of a guard against this. Based on field tests of Free Haven, we may need to switch to a policy of ignoring referrals which do not have receipts.

5.6 Entrapment

There are several ways in which an adversary can appear to violate the protocols. When someone points this out, the adversary can present receipts which show him wrong and accuse him of the above attack.

There is no defense in the present implementation against this attack; a more thorough system of attestations and protests is necessary.

Chapter 6

Future Works

There are several areas in which this Trust Module falls short. The Free Haven group hopes to resolve most of them over the next few months, as we work towards our first public release. Others require advances in the theory of cryptographic anonymity:

Trust Algebra There is no good system for formally and rigorously talking about trust and approval. We need one in order to be certain that the trust system does anything like what we are hoping for.

Entrapment Defense The current belief in referrals is somewhat naïve. We need better support for logging various referrals and integrating their results. The best approach is beginning to look like some form of linear algebra to deal with the loops of trust and metatrust.

Deployment Many of the questions we have about Free Haven and the Trust Module cannot be answered with thought experiments. We need to get the system deployed and working in a test environment to see where it is attacked and where it breaks. This test system needs to be clearly marked as potentially unsafe.

Chapter 7

Conclusions

Wow. Building trust systems is hard.

— *Roger Dingledine*

I remain convinced that the economic model of trust offers valuable insights, and will eventually lead to a rigorous model of anonymous trust. What exists now is a framework for testing various assumptions about trust and anonymity. With the experience of an active Free Haven, it will be able to grow into a more robust system.

For now, the Free Haven relies in large part on goodwill and generosity of the community to provide resources to ensure that there is sufficient protection against corrupt or malicious nodes.

Appendix A

Acknowledgements

This paper was written with the help and advice of several people to whom I am particularly grateful. They include:

- Roger Dingledine provided the inspiration for the Free Haven Project and coordinated the efforts of all involved. Late-night discussions with him on the nature of anonymity, trust and confidence inspired the trust economy concept.
- Seph Sokol-Margolis, David Molnar, Michael Freedman, and the rest of the Free Haven Group provided many hours of valuable discussion on the merits and flaws of this system/
- Susan Born caused this document to be readable by non-cryptographers and readers of formal English.
- Professor Ron Rivest, as thesis supervisor to Roger, Michael, and myself, contributed greatly to our online discussions and provided a dose of political realism.

Bibliography

[And] Ross Anderson. The Eternity Service. http://www.cl.cam.ac.uk/users/rja14/eternity/eternity.html.

[AOL] Aol instant messenger. http://www.aol.com/aim.

[Din] Roger Dingledine. The Free Haven Project. http://freehaven.net/.

[Din00] Roger Dingledine. The Free Haven Project. Master's thesis, MIT, 2000.

[ea] Ian Hall-Beyer et al. Gnutella. http://gnutella.wego.com/.

[Hel] Johan Helsingius. press release announcing closure of anon.penet.fi. http://www.penet.fi/press-english.html.

[KBD98] Clifford Kahn, David Black, and Paul Dale. MANET: Mobile agents for network trust. http://www.darpa.mil/ito/psum1998/F255-0.html, 1998.

[Kro] Jason Kroll. Crackers and crackdowns. http://www2.linuxjournal.com/articles/culture/007.html.

[Lew] Mark Lewis. Metallica sues Napster, universities, citing copyright infringement and RICO violations. http://www.livedaily.com/archive/2000/2k04/wk2/MetallicaSuesNapster,Univ.html.

[Mil67] Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[Nap] Napster. http://www.napster.com/.

[PGP] PGP FAQ. http://www.faqs.org/faqs/pgp-faq/.

[tC] the Cleaner. Gnutella wall of shame. http://www.zeropaid.com/busted/.

[WRC] Marc Waldman, Aviel Rubin, and Lorrie Cranor. Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system.

[zks] Zero Knowledge Systems. http://www.freedom.net/.