Attack resistant trust metrics

by

Raphael L. Levien

M.S. (University of California at Berkeley) 1995

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Alexander Aiken, Chair
Professor Jasper Rine
Professor Joseph Hellerstein

2002

The dissertation of Raphael L. Levien is approved:

Chair Date

Date

Date

University of California at Berkeley

2002

Attack resistant trust metrics

Abstract

Attack resistant trust metrics

by

Raphael L. Levien

Doctor of Philosophy in Computer Science

University of California at Berkeley

Professor Alexander Aiken, Chair

This dissertation characterizes the space of trust metrics, under both the scalar assumption where each assertion is evaluated independently, and the group assumption where a group of assertions are evaluated in tandem. We present a quantitative framework for evaluating the attack resistance of trust metrics, and give examples of trust metrics that are within a small factor of optimum compared to theoretical upper bounds. We discuss experiences with a real-world deployment of a group trust metric, the Advogato website. Finally, we explore possible applications of attack resistant trust metrics, including using it as to build a distributed name server, verifying metadata in peer-to-peer networks such as music sharing systems, and a proposal for highly spam resistant e-mail delivery.

Professor Alexander Aiken
Dissertation Committee Chair

Dedicated to the memory of

Prof. Eugene Lawler

Contents

List of Figures **vi**

List of Tables **vii**

I First Part **1**

1 Introduction **2**

2 Trust Metrics **6**
- 2.1 The simplest trust metric . . . 6
- 2.2 Survey of the literature . . . 7
- 2.3 Framework for analysis . . . 9
- 2.4 Analysis: upper bounds . . . 10
- 2.5 Analysis: lower bounds . . . 11
- 2.6 Analysis of Maurer's trust metric . . . 11

3 Group Trust Metrics **12**

4 Advogato **13**
- 4.1 Implementation overview . . . 13
- 4.2 Implementation details . . . 14
- 4.3 Experimental results . . . 16

5 An attack-resistant name service **17**
- 5.1 Policy language . . . 18
- 5.2 Implementation . . . 18

6 Attack-Resistant Metadata **20**
- 6.1 Brief analysis of PageRank . . . 21
  - 6.1.1 Recap of PageRank . . . 21
  - 6.1.2 Attack-resistance of PageRank . . . 22
  - 6.1.3 Significance of initial vector and walk length . . . 23

6.2 What makes a trust metric attack resistant? 25
6.3 Generalized metadata 26
6.4 Distributed network model 27

7 Stamp Trading Networks **29**
7.1 Stamps 30
7.2 Reputation policy 32
7.3 Applications 32
7.4 Open questions 32

Bibliography **33**

List of Figures

List of Tables

Acknowledgements

Thanks foremost to my family for supporting me through the ordeal of completing a dissertation.

Thanks to the #gimp crew for stuff to do while I was bored with the thesis, and for being true friends through rough times as well as good times.

Thanks to Miles Jones for giving me the opportunity and time to complete this dissertation.

Thanks to Roger Dingledine for restoring my faith in the academic community, especially after a number of papers had been rejected.

Thanks to Bryce Wilcox-O'Hearn for being Zooko.

Finally, thanks to 100mg/day of Sertraline HCl, for letting me keep my head above water.

Part I

First Part

Chapter 1

Introduction

In today's world of open, decentralized networks, the question of *trust* is becoming increasingly relevant. Most existing Internet protocols implement a naive policy of providing a relatively limited set of services, but trusting all users with them. Given that a significant fraction of Internet users are not trustworthy, the inevitable result is spam, denial of service attacks, cracking, and a host of other ills far too familiar to legitimate Internet users.

A number of approaches have been developed and deployed over the years, with varying degrees of success. The most basic is the model of password-protected accounts. This mechanism is almost universally deployed, but suffers from serious limitations, including the need for servers to manage the accounts, the need for users to keep track of a large number of passwords, and the relative lack of security provided by this model. Thus, there has been a sustained interest in more sophisticated models.

One such approach is the *Public Key Infrastructure*, or PKI [5]. Briefly, a PKI consists of various *Certification Authorities* (or CA's) that issue *certificates* asserting a binding between a name and a public key. PKI's suffer from two fundamental problems: the lack of useful meaning in the PKI's underlying namespace, and the question of which CA to trust. Actual implementations of CA's have proved themselves not worthy of absolute trust. Further, as the number of CA's deployed scales up, the risk of any one of them being compromised scales accordingly. In part because of these two problems, PKI's have met with limited success at best.

Spurred on by these limitations, much recent attention has been focussed on the explicit encoding and evaluation of trust relationships. A pioneering work in this area is the PolicyMaker framework of Matt Blaze[4]. A particularly interesting branch of this work is the concept of *trust metrics,* which are the primary focus of this thesis.

Trust metrics are based on the principle of *local* encoding of trust relationships, but granting trust on a global scale. These assumptions closely parallel the philosophy of peer to peer networks[17].

While the literature is rich with design proposals for trust metrics, there has been relatively little analysis of how well they work in practice. In Chapter 2, we show that, under assumptions similar to the Internet, the entire category of "scalar" trust metrics fails to resist easily-mounted attacks.

While the outlook for scalar trust metrics is indeed bleak, we propose a new class, which we call *group* trust metrics, so called both because the trust metric is very well suited to evaluating membership in a group, and because this evaluation is done over the entire group of nodes, rather than individually for each node. While the group trust metric cannot prevent individual hostile nodes from being accepted as group members, it can place strict bounds on the *number* of hostile nodes so accepted.

I built a community website for free software developers, called Advogato, that uses this group trust metric to determine who belongs to the community. Acceptance by the trust metric confers privileges to post articles and comments, and to edit project information. At the time of this writing, the site has been in operation for about 18 months, and has built a trust graph of over 1000 active nodes. Advogato is notable for the extremely low level of trolls, spam, and other forms of abuse common to bulletin board type systems, thus providing strong anecdotal evidence that the trust metric is effective. Advogato is described in more detail in Chapter 4.

One interesting application of trust metrics is secure registration and lookup of public keys bound to names, essentially the same problem addressed by Public Key Infrastructures (PKI). The attack resistant properties of the trust metric avoid the single point of vulnerability common to most PKI designs. We present a detailed design for such an attack-resistant name service in Chapter 5. The desire to build a "better PKI" was the original motivation behind the work of this thesis.

Because the group trust metric is effective in resisting attack, it has many other interesting applications. One such is an attack-resistant system for distributing metadata, for example opinions about songs. Attack resistance can be useful for ensuring high quality of the metadata, but becomes particularly important when there are financial implications to the metadata system, for example to identify recipients of voluntary donations to artists who post their music to the Net. We propose a design for an attack resistant metadata system in Chapter 6. This design combines ideas from both the Advogato group trust metric and the PageRank algorithm used in Google.

Another intriguing application for group trust metrics is to provide a spam resistant infrastructure for e-mail. Spam is a huge problem, causing lots of wasted time for many, and most e-mail users feel hopeless in the face of it. We present a partial design for a communications infrastructure resistant to spam, yet highly permeable to legitimate email. This design uses a capacity constrained flow network where "stamps" are the commodity of flow. Rather than explicitly computing a flow in the network, the flow results from local activity. When capacity is exhausted, e-mail is no longer deliverable. This design has many appealing properties, but several aspects remain as open problems, particularly scaling. Chapter 7 presents this design and discusses some open issues.

Chapter 2

Trust Metrics

In this chapter, we review the literature of trust metrics, present a quantitative framework for analyzing the attack resistance of trust metrics. Finally, we prove tight upper and lower bounds on the attack resistance of trust metrics given the usual scalar assumptions.

2.1 The simplest trust metric

In this section, we present the simplest possible trust metric, which forms the basis of other trust metrics in the literature.

There are three inputs to this trust metric: a directed graph, a designated "seed" node indicating the root of trust, and a "target" node. We wish to determine whether the target node is trustworthy.

Each edge from s to t in the graph indicates that s believes that t is trustworthy. The simplest possible trust metric evaluates whether t is reachable from s. If not, there is no reason to believe that t is trustworthy, given the data available.

In a cryptographic implementation, each node corresponds to a public key, and each edge from s to t corresponds to a digitally signed *certificate*. In the usual terminology, s is the *issuer*, t is the *subject*. The certificate itself is some string identifying t, along with a digital signature of this string generated by s.

The simplest trust metric is also the weakest. If an attacker is able to generate an edge from any node reachable from the seed to a node under his control, then he can cause arbitrary nodes to be accepted. As the size of the reachable subgraph increases, the risk of any such attack increases as well. Thus, the primary focus in the literature is to present stronger trust metrics that (hopefully) resist attacks better, while still accepting most nodes that deserve trust.

Section 2.2, discusses a sampling of trust metrics previously published. Section 2.3, presents a quantitative notion of "attack resistance."

2.2 Survey of the literature

All trust metrics include the three inputs described above: the trust graph, the seed node, or *trust root*, and the target. Some trust metrics add more detail to these inputs. Trust edges may contain some conditions or restrictions, for example that t is himself trustworthy, but cannot be trusted to vouch for other nodes. In addition, the target may be a richer assertion than merely whether a certain node is trustworthy. For example, edges may additionally identify a maximum dollar value, and the target may be an assertion that the target node can be trusted with a transaction of some dollar value.

Finally, there may be additional policies to be enforced by the trust metric, or parameters supplied. Often, these parameters control the overall strictness or tolerance of the trust metric.

The most general form of "trust management" system is represented by PolicyMaker[4], as certificates and policies can represent arbitrary computations in Turing-complete language. Applications of PolicyMaker tend to focus on the language of assertions rather than trust computations over the graph, but the fully general nature of the system allows the the latter to be implemented.

Monotonicity assumption: consistent with supplicant presenting signed certificates to verifier. Relationship to scaling: verifier only has access to small subset of entire trust graph.

Another relatively simple trust metric is similar to the simplest one, with the additional constraint that path lengths are bounded by some parameter k. Thus, all nodes within a distance of k edges from the seed are accepted. A variation of this trust metric is used in X.509 systems.

Perhaps the earliest published system resembling a modern trust metric is the Beth, Borcherding, and Klein[2]. This system contains a set of elaborate inference rules for deriving a "trustworthiness" value between 0 and 1, using both "direct trust" and "recommendation trust" relationships encoded on edges. However, further work by Reiter and Stubblebine[22] showed this trust metric to be no more secure than the simple reachability criterion described above. Note that the BBK trust metric is not monotonic.

A still earlier system is described by Tarah and Huitema[25], who suggest evaluating both certificates and "certificate paths." They suggest several possible simple trust metrics, including the length of the certification path and the minimum of involved trust values along the path. However, because it evaluates only paths, rather than general graphs.

Reiter and Stubblebine[21] presented the first trust metric with the ability to resist nontrivial attacks. Briefly, this trust metric counts the number of node-independent paths of bounded length from the seed to the target. Thus, both the path length and the threshold for the minimum number of such paths are tunable parameters of the metric. A simple characterization of its attack resistance follows easily: an attacker must add "bad" edges to nodes under his own control to at least as many nodes as the threshold for the number of independent paths. Even though evaluation of this trust metric is an NP-complete problem, the experimental PathServer implementation seems to perform reasonably well in practice.

Maurer presented an intriguing trust metric based on randomized experiments[16]. Very briefly, each edge in the network has an associated probability. The result of the trust metric is the probability that the target is reachable from the seed in a subgraph of the original graph where each node is present with the probability given in the original graph.

Both implementation and analysis of the Maurer trust metric are challenging. However, we present an analysis below (Section 2.6) suggesting that fails to resist any non-trivial attack. To do so, we will first need to prepare a framework for quantitatively evaluating the attack resistance of a trust metric.

2.3 Framework for analysis

What does it mean to be attack resistant?

Given an attack of a certain cost, how many bad assertions will be accepted?

How to quantify cost of attack?

Node attack: number of nodes for which attacker can make arbitrary certs. Corresponds to stealing the public key of victim.

Edge attack: number of certs. Corresponds to fooling the victim once.

Protection against node attack is a stronger property than protection against edge attack. Attack of a single node can correspond to an arbitrary number of edges.

Another factor in trust metric: how many certs are needed? At simplest, indegree of target node.

Flavors of attack: random victims, chosen victims.

2.4 Analysis: upper bounds

Strict upper bounds on effectiveness of trust metric. For node attack, if n nodes are attacked, entire system falls, where n is minimum indegree of target node.

Proof: attacker chooses target that is accepted (with minimum indegree), and chooses its predecessors as victims. Since indegree is n, attack is n nodes. Removing the original target, the graph is identical to the one in which the original target is accepted, so the trust metric can't distinguish.

For edge attack, if n^2 nodes are attacked, entire system fails. Choose target as above, then spoof inedges of predecessors of target.

2.5 Analysis: lower bounds

This section analyzes existing trust metrics.

Present simple flow-based metrics, and prove (using min-cut) that they almost exactly meet the upper bounds given above.

2.6 Analysis of Maurer's trust metric

Analysis of Maurer: follow [15].

Chapter 3

Group Trust Metrics

This chapter presents the concept of *group trust metrics,* which have significantly better attack resistance properties than trust metrics under the scalar assumption.

Central idea: compute trust metric over entire graph at once.

Thus, no monotonicity for individual assertions. In fact, acceptance of individual assertions is nondeterministic.

However, a weaker monotonicity property: the number of assertions accepted is monotonic with the graph.

Capacity constrained flow network.

Capacities of nodes are set as a function of distance from seed.

Cite Self-Organization paper[10].

Chapter 4

Advogato

We have implemented the group trust metric described in Chapter 3, and deployed it as the key part of the Advogato website. In this chapter, we describe the implementation, analyze the data collected, and present some observations on lessons learned.

The Advogato website launched in November 1999, and has been in continuous operation since. It serves as a community message board for free software developers. The heart of Advogato is the certification mechanism, essentially a straightforward implementation of the group trust metric.

4.1 Implementation overview

Advogato is implemented as a C-language Apache module, available under the GNU General Public License.

Internally, mod_virgule is heavily based on XML. All of the account information (including the peer certifications), diary entries, articles, responses, and project information is stored in the filesystem in XML format. To handle a request, mod_virgule dynamically generates HTML from these XML files. The content varies slightly if the user is logged in, as determined by the standard HTTP cookie mechanism.

Advogato extends the yes/no evaluation of the trust metric presented in Chapter 3 to three levels, entitled Apprentice, Journeyer, Master. Each certification edge has an associated level. The underlying trust metric is run once for each level. For a given level, edges below that level are discarded. For example, to compute the Journeyer level, the underlying trust metric is run on the subset of the trust graph consisting of Journeyer and Master edges. The final certification assigned to the node is the maximum level for which the node is accepted.

Trust "seed" is hard-wired. In practice, hard-wired seed hasn't caused problems, but it is a concern. Providing user-configured alternate seeds is a possible extension, but the need for it hasn't been strongly felt.

4.2 Implementation details

One motivation for the implementation choices was to experiment with the platform provided by the Apache module interface. C is not a popular language for implementing Web sites. Yet, our experience has been quite good.

A core feature of the Apache platform is the use of resource *pools*. In a pool discipline, memory and other resources are allocated from an explicitly identified pool. At some later time, when all memory allocated in the pool is known to be unreachable, the entire pool is freed.

Pools have been used for a long time. Ross[23] presented a storage package with explicitly identified *zones*, roughly equivalent to Apache's pools. There has been a fair amount of recent interest in providing explicit language support for pools, including extensions to the C language[11], theoretical language designs[7], and the ML Kit compiler for Standard ML[26].

In the case of Apache, there is no explicit language support. Rather, allocation functions take an additional explicit pool argument, and are called using the standard C runtime discipline. For example, the low level memory allocation function is:

void *ap_palloc (struct pool *p, int nbytes)

In the context of mod_virgule, most allocation is done in a *request pool*, which has a lifetime exactly equal to one Web request. When the request is complete, the Apache process frees the request pool, and typically becomes available to serve the next request.

The pool discipline is appealing in many ways. When properly used, memory management is nearly painless. Further, memory leaks are rare. These advantages are similar to those provided by garbage collecting runtimes, but with several advantages:

- Finalization is deterministic.
- There is no need for language or runtime support.
- Time and space overhead is minimal.
- The discipline interoperates easily with external libraries.

The last of these points is demonstrated by mod_virgule's use of libxml[**?**].

4.3 Experimental results

Provide data about graph: number of active nodes in trust graph (¿ 1000), number of Apprentice, Journeyer, and Master nodes. Also present distribution of capacities of nodes. Average capacity of accepted nodes is related to vulnerability of trust metric in the face of random attack.

Maybe visualize Master sub-graph, although it's already too tangled to see much.

One conclusion: manually creating certifications to build a trust graph is realistic, if there is some expected payoff.

Chapter 5

An attack-resistant name service

This chapter presents the design of an attack-resistant name service, based on the group trust metric described in Chapter 3.

Name service is a thorny problem: as much social as technical. Real world CA's have misbehaved.

We resolve these issues by separating mechanism from policy. This requires policy to be formally specified. Thus, the policy language must be rich and flexible enough to accommodate the policies desired.

Probably the only workable policy is "first-come, first-served," as it's the only one that's understandable.

Choice of namespaces: flat, hierarchical, hash-based. Also cite SDSI for relative namespaces.

5.1 Policy language

Goal is DNS-like hierarchy with some flexibility.

A specific goal is that compromise of a name in the hierarchy doesn't immediately lead to compromise of entire subtree.

Use policy constraint language to balance power of "owners" of name in hierarchy with expectation of subnames to not be mutable.

Evaluation of policy wrt policy constraint language is co-NP complete.

Present pseudocode algorithm for implementing policy constraint language.

5.2 Implementation

Peer to peer network.

Square-root scaling (contrast with Chord[24]).

Global knowledge of all servers. Join/leave protocol is beyond the scope of this design, but cite Gnutella, Mojo Nation metatrackers, and Chord.

Each node is individually responsible for enforcing the policy.

Spread name to "responsible subset" of all servers, using hash function.

Use group trust metric to evaluate assertions of the form "node n is a trustworthy node."

When validating a name/key binding, query all trustworthy servers, and vote.

Thus, a bound on the fraction of bad servers bounds the probability of accepting a bad name/key binding.

No consistency guarantees (but combining with [6] ideas is interesting research topic).

Chapter 6

Attack-Resistant Metadata

Systems should be designed so that metadata is attack-resistant, as well.

One important form of metadata: "this song is beautiful."

Another example: micropayments in a music distribution system, commonly known as "pay Lars." But: which Lars? Cite Scott McCloud's discussion of micropayment systems for comics as well.

Another form of metadata that has received substantial recent attention is relevance of Web search results. To answer this query, an extremely rich trove of raw data is available: the link structure of the several billion documents currently available on the World Wide Web. Two systems have attracted particular attention: IBM's Clever[13], and the PageRank algorithm used by Google[18]. The latter is claimed to have some attack resistant properties, and the experience from Google's deployment is encouraging in this regard: the relevance of Google's search results is widely praised, and it does seem to be rather difficult for spammers or others wishing to unfairly dominate the search results to do so, except at considerable expense.

One interesting recent example is the dominance of pro-Scientology listings in the Google results for a search on the keyword "scientology." This is discussed in an online essay[20]. To Google's credit, as of 2 Mar 2002, this search yields one anti-Scientology site, in fact at #4. *(probably cut this—the scieno thing has obviously become far more a political issue than a technical one)*

6.1 Brief analysis of PageRank

A description of the PageRank algorithm was published in 1998[18], but as the work has continued as a business rather than an academic pursuit, there has been no published followup. This section attempts to address this hole in the literature, by providing a brief discussion of the PageRank's attack-resistance.

6.1.1 Recap of PageRank

For the purposes of analyzing the PageRank algorithm, it is most useful to present it in its random walk formulation. Each walk begins by choosing a node u, using an initial distribution $E(u)$. At each step, with probability $||E||_1$, the walk ends. Otherwise, the next node in the walk is chosen uniformly from the successors. Thus, the probability that a walk is of length k is $||E||_1(1 - ||E||_1)^k$. The rank assigned to each node is the probability that such a walk will end at that node.

Here, $||E||_1$ denotes the L_1 norm of E, informally "Manhattan distance" and formally:

$$||E||_1 = \sum_u |E(u)|$$

Note that when E is nonnegative, $||E||_1 = E \cdot \mathbf{1}$, where $\mathbf{1}$ is the vector of all ones.

The PageRank paper also presents an eigenvector formulation of the algorithm. The adjacency graph A is defined as follows: A_i, j is $1/N_i$ if there is an edge from i to j, otherwise 0, where N_i is the outdegree of node i. Let $R = c(A + E \times \mathbf{1})R$, where $||R||_1 = 1$, and c is maximized. Thus, R is an eigenvector of $(A + E \times \mathbf{1})$. The eigenvector formulation is equivalent to the random walk formulation: $R(u)$ is equal to the probability that a random walk ends in node u.

6.1.2 Attack-resistance of PageRank

What exactly do we mean to say that an algorithm such as PageRank is attack resistant? In fact, the criteria are basically identical to the Advogato group trust metric presented in Chapter 3. The input to the PageRank algorithm is a graph. In the Google application of PageRank, nodes correspond to web pages, and edges correspond to links. The "seed" of the Advogato trust metric corresponds to the initial vector E of PageRank. Finally, the output of PageRank is a real-valued ranking, while Advogato produces a boolean accept/reject value for each node. Note that the Advogato trust metric can be extended to multiple levels of trust by doing multiple runs of the algorithm with different capacities.

There are a number of different ways to quantify the success of an attacker. Here, largely for purposes of simplifying the proof outline, we choose the sum of the rank accorded to all nodes under the attacker's control. This measure is very similar to the one chosen for the analysis of the Advogato trust metric: the number of "bad" nodes chosen.

We state the attack resistance property of PageRank more formally. For any labelling of the nodes in the graph G into "good" and "bad" nodes, the sum of the rank of the "bad" nodes is bounded as follows: ...

The analysis of the attack-resistance of PageRank is dependent on the choice of the initial source of rank, $E(u)$. As in **??**, let the nodes in the "trust graph" (ie web pages) be divided into three classes: good, bad, and confused. To recap, a "confused" node is one that is itself good, but may contain links to bad nodes. Let us then assume that $E(u)$ is zero for all bad nodes u.

Now, it is possible to see that the resulting PageRank meets the "bottleneck" criterion of **??**. A given walk will lead either to a good page or a bad page. In the worst case, if the walk ever touches a bad node, then the resulting page may be bad. Otherwise, it is good.

6.1.3 Significance of initial vector and walk length

Here, we see that the choice of initial vector, as in the choice of "seed" in the Advogato trust metric, has a critical effect on the quality of the results.

If the fraction of good nodes in E is p, then the total rank accorded to good nodes is multiplied by p.

One of the choices of E presented in the PageRank paper is a uniform distribution over the set of all nodes in the graph. This value for E is not attack-resistant; an attacker can simply create a huge number of web pages, and have them link to a "bad" page which will then receive a high rank.

Thus, to achieve attack resistance, E must be chosen with some tendency toward good nodes. Obviously, if it is possible to do this with perfect accuracy, the trust metric is not needed. In general, a good strategy is to hand-pick a small subset of sites that are known to be good, and let the walks cover the rest of the graph. In the context of websites, one good way to choose start nodes is to use a collaboratively edited link directory such as the ODP (dmoz.org).

Given this assumption, the choice of walk length becomes clearer. The shorter the walks, the more attack resistant. However, if walks are too short, then good nodes are inadequately reachable from the start nodes in E. The PageRank paper gives a value of 0.15 that the walk ends at each step. Thus, walk lengths are an exponential distribution with a mean of $(1 - ||E||_1)/||E||_1$, thus 5.67 for $||E||_1 = 0.15$. This probably represents a good real-world balance between complete coverage and low vulnerability.

The PageRank paper presents another intriguing choice for E: the set of "root" web pages. This set can be determined automatically, reducing the dependence on manual labor. Since setting up a root web site requires purchase of a domain name from a DNS cartel, typically on the order of $10 per year, mounting an attack requires money, generally in an amount proportional to the sum of rank desired. It is unclear whether Google uses this criterion for their own E. If so, this analysis lends credence to the widespread belief that cross-linked web sites with many different domain names are a good way to increase Google rank.

6.2 What makes a trust metric attack resistant?

Now that we know of two different trust metrics that we consider attack resistant, it is worth investigating what the similarities and differences are.

Network flow and principal eigenvectors are two different computations. They are not used interchangeably. However, there is a common thread which allows both algorithms to serve as the basis of attack resistant trust metrics. In this section, we propose the *bottleneck property* as a common feature of both trust metrics. We conjecture that this property is satisfied by all attack resistant trust metrics, although at there is no proof.

The bottleneck property, informally stated, is that the total trust quantity accorded to an $s \to t$ edge is not significantly affected by changes to the successors of t. Informally, when s is a "good" node and t is a "bad" node, this means that manipulation on the part of the "bad" nodes does not affect the trust value.

Scalar trust metrics, no matter how carefully tuned, all fail to satisfy the bottleneck property, and are not attack resistant. It is fairly easy to see why they do not: since the trust computation is performed independently for each final target node, simply adding more target nodes increases the total amount of trust accorded to target nodes. To be attack resistant, a trust metric must dilute the trust accorded to the successors of t as more successors are added, either by decreasing the value of each (as in the case of PageRank), or decreasing the relative probability that any one of these successors is accepted (as in Advogato).

The class of attack resistant trust metrics may have more instances than Advogato and PageRank. Network flow and eigenvectors are both compelling in their simplicity, resulting in relatively easy analysis and implementation. Even so, it may be that some other, yet to be discovered, trust metric is better suited to real applications.

6.3 Generalized metadata

The results of a trust metric or Web page ranking represent a fairly special class of metadata: essentially boolean or real "confidence value" for inclusion in a group. This section describes how an eigenvalue-based trust metric might be extended to more general forms of metadata assertions.

The primary new concept is that of an "assertion," which is simply a statement of some sort. Examples of meaningful assertions include "I think the song hashing to `01..EF` rates 9 out of 10", "Tips to be paid to the artist of song `01..EF` should be paid to Swiss bank account 12345", "The song hashing to `01..EF` is actually *Man of Constant Sorrow*, recorded by the Soggy Bottom Boys".

The ranking of such pieces of metadata is given in terms of random walks. Begin the walk at a start node (generally, this will correspond to the user of the ranking system). At each step, if that node contains a matching metadata record, the walk stops. If not, then the walk with some fixed probability p terminates, or proceeds to a successor chosen at random.

The concept of "matching" metadata record requires a bit of further elaboration. For example, it makes little sense to rate a song as both 4 out of 10 and 9 out of 10. Thus, the template for such a query is "I believe song `01..EF` ranks * out of 10". The random walk will end at any node containing such a matching record.

6.4 Distributed network model

This section presents a distributed network model of the eigenvector approach. The initial presentation is based on PageRank, but it extends straighforwardly to more general metadata.

Each node i maintains a vector $R_i(j)$, assigning a real value to each other node j, with $||R_i||_1 = 1$. Periodically (but asynchronously), each node i performs an *update* operation, as follows. For each predecessor node s in the trust graph, node i requests the current value of R_s from node s. All such vectors must satisfy $||R_s|| = 1$; if not, the vector is clearly bad and should be discarded. A new R_i is recomputed as follows:

$$R_i(j) = \frac{(1-p)}{|\mathrm{pred}(i)|} \sum_{s \in \mathrm{pred}(i)} R_s(j) + (p \text{ if } i = j, 0 \text{ otherwise})$$

It should be clear that such a network will converge to the same value as the random walk and eigenvector formulations as presented above. Specifically, $R_i(j)$ is the probability that a random walk beginning at i will end at j, with p representing the probability that the walk terminates at each step.

Thus, this network computes the "personalized PageRank" for each node in the network. Distributing the computation may ease the computation and communication bandwidth bottlenecks inherent in a centralized approach.

Further, the distributed model eliminates the need for global knowledge of the trust graph. Each node need only query its immediate predecessors.

To adapt this network model to generalized metadata as described in Section 6.3, R_i becomes a vector over assertions, rather than over other nodes. For all metadata statements j local to node i, $R_i(j) = 1$, and $R_i(k) = 0$ for all $k \neq j$ and for which the templates of j and k match. For example, if j is "Song X ranks 7 out of 10", then statements "Song X ranks 4 out of 10" receive a 0 value. For all metadata statements not covered by local assertions, the propagation formula above is used.

In actual implementation, scaling becomes a problem as the number of elements in the vector R grows. For a small network, or even a large network covering a relatively small number of metadata statements, scaling may not be a serious problem. Some techniques worth exploring include:

- Pruning the vector for near-zero values.
- Data compression of the vector.
- Adaptive transmission, so that slow-changing values of $R_i(j)$ are sent less frequently.

Analysis: why is such a network attack resistant? The "bottleneck property" generalizes the capacity constrained flow network of the group trust metric presented in Chapter 3.

Discussion of dynamic behavior: can optimize for "diamond in the rough", by selecting for high recommendations with low confidence value. Then listen to song and inject a new recommendation into the network. If song is excellent, then recommendations will propagate quickly through the network. If song sucks, then low recommendations will remove song from the diamond in the rough category. The goal is to get good coverage of reviews while minimizing the number of people forced to listen to mediocre songs.

Chapter 7

Stamp Trading Networks

This chapter describes how capacity constrained flow networks may be used to build a scalable and robust peer to peer network infrastructure. Peer to peer networks have to deal with the following questions, usually much more straightforward in centralized networks:

- How can a node discover other nodes to communicate with?
- If there is a subset of nodes offering a particular service, how can a node discover nodes in that subset?
- How can scarce resources (storage and bandwidth) be allocated in an open network, where services are provided to all peers?

These questions are so difficult that most implementations of peer-to-peer networks decentralize only part of the total network architecture (for example, movement of bulk data), while retaining centralized servers for this "network metadata." Only very innovative designs such as Freenet[?] and Gnutella[?] embrace a completely decentralized approach, and such designs tend to suffer from both scalability problems and vulnerability to malicious attacks. For example, there is discussion of "cancerous nodes" in the Freenet community.

This chapter describes a partial design for a peer to peer network in which the discovery of other nodes, as well as the provisioning of services, is governed by a flow of "stamps," or simple authentication tokens. The initial flow of stamps is determined by manually configured trust relationships, similar to those of Advogato and the other trust metrics described in this thesis. The flow of stamps is effectively a capacity-constrained network, thus providing the network as a whole with some attack resistance.

The design presented in this chapter is somewhat speculative. It has not yet been implemented, even as a prototype, and there is no doubt a considerable amount of refinement needed to become truly practical. Even so, it represents a dramatic improvement in robustness over any peer to peer network design currently known, and at a level of efficiency rivalling that of the most advanced designs.

7.1 Stamps

We assume a peer-to-peer network of *nodes* on the Internet. Each node is implemented as a daemon process with access to storage for the node's state. Further, each node has an associated public/private key pair. The hash of the node's public key serves as its id. Each node listens for connections on an IP address and port, and can request connections to other IP address/port tuples. Such a model is entirely standard, and is shared by numerous other peer-to-peer networks, including MNet[27], Freenet[?], TODO.

Peer to peer networks can be built by issuing and trading "stamps".

Each node issues its own stamps, and gives them at a steady rate to its trust neighbors.

A node performs services (for example, serving files) in exchange for its own stamps.

Nodes also keep stamps on hand for other nodes in the system. A node will trade these other stamps in exchange for some of its own.

Thus, movement of stamps through the network is a capacity constrained flow network. This limits the amount of services an attacker can consume by the number of edges to bad nodes.

How to determine which other stamps to keep on hand?

One choice is to try to obtain stamps for all other nodes in the network. This won't scale.

Trying to find stamps for a given node is essentially a routing problem. Consider the "stamp graph" induced as follows: an edge from s to t exists if s has an adequate number of t's stamps on hand. Given a route from s to x, s can walk that graph, at each node trading the stamps it has for those of the next node in the graph.

How to route when global information about the graph is not available? This is an interesting design problem, beyond the scope of this thesis (dammit!). Chord[24] contains ideas on how such a network may be made scalable.

7.2 Reputation policy

7.3 Applications

Potential application for stamp trading: spam-resistant email. One variation: whether to ring the target's cell phone or go to voicemail.

7.4 Open questions

The design presented in this chapter is quite speculative. There are two large open questions that should probably be answered before the design is implemented. First, can the system efficiently and reliably find short routes through the network? Second, is the reputation policy described in Section 7.2 sufficient to prevent against false-negative attacks?

These questions remain as directions for future research.

Bibliography

[1] Albert-Laázló, Réka Albert, and Hawoong Jeong. Scale-free characteristics of random netwoks: The topology of the world wide web. *submitted to Elsevier Preprint*, August 1999.

[2] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. *Lecture Notes in Computer Science*, 875:3–??, 1994.

[3] Matt Blaze. Oblivious key escrow. In *Proc. Workshop on Information Hiding*, number 1174 in LNCS, pages 334–343. Springer-Verlag, 1996.

[4] Matt Blaze, Joan Feigenbaum, and J. Lacy. Decentralized trust management. In *Proc. 17th Symposium on Security and Privacy*, pages 164–173, Los Alamitos, 1996. IEEE Computer Society Press.

[5] Marc Branchaud. A survey of public key infrastructures. Master's thesis, McGill University, Dept. of Computer Science, March 1997.

[6] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proc. Third Symp. on Operating Systems Design and Implementation*, New Orleans, February 1999.

[7] Karl Crary, David Walker, and Greg Morrisett. Typed memory management in a calculus of capabilities. In *Conference Record of POPL 99: The 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, Texas*, pages 262–275, New York, NY, 1999.

[8] Frank Dabek, Emma Brunskill, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica, and Hari Balakrishnan. Building peer-to-peer systems with chord, a distributed lookup service. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Schloss Elmau, Germany, May 2001. IEEE Computer Society.

[9] Roger Dingledine and Paul Syverson. Reliable MIX cascade networks through reputation. In *Proc. Financial Cryptography*, March 2002.

[10] Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans Coetzee. Self-organization of the web and identification of communities. *IEEE Computer*, 35(3):66–71, 2002.

[11] David Gay and Alexander Aiken. Language support for regions. In *SIGPLAN Conference on Programming Language Design and Implementation*, pages 70–80, 2001.

[12] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd ACM Symp. on Theory of Computing*, 2000.

[13] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[14] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[15] Raph Levien and Alexander Aiken. Attack resistant trust metrics for public key certification. In *7th USENIX Security Symposium*, San Antonio, Texas, January 1998.

[16] Ueli Maurer. Modelling a public-key infrastructure. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Computer Security – ESORICS '96*, number 1146 in LNCS. Springer Verlag, 1996.

[17] Andy Oram, editor. *Peer to Peer.* O'Reilly & Associates, 2001.

[18] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[19] David Post, 1999.

[20] ptsc (pseudonym). The Church of Scientology's supremacy over the search term "Scientology" on Google, February 2002.

[21] Michael Reiter and Stuart Stubblebine. Path independence for authentication in large-scale systems. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 1997.

[22] Michael Reiter and Stuart Stubblebine. Toward acceptable metrics of authentication. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, 1997.

[23] D. T. Ross. The AED free storage package. *Communications of the ACM*, 10(8):481–492, August 1967.

[24] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.

[25] Anas Tarah and Christian Huitema. Associating metrics to certification paths. In *European Symposium on Research in Computer Security (ESORICS)*, pages 175–192, 1992.

[26] Mads Tofte and Jean-Pierre Talpin. Region-based memory management. *Information and Computation*, 1997.

[27] Bryce Wilcox-O'Hearn. Experiences deploying a large-scale emergent network. In *Proc. 1st International Workshop on Peer-to-Peer Systems*, March 2002.