Data Mining Application: Attack Resistant Trust Metrics

Mrs. Aparna Atul Junnarkar[1], Dr. Udai singh Sutar[2], Mr. Amol Rajmane[3]

[1]Sr. Lecturer (Computer Engineering)-- Modern College of Engineering , Shivajinagar, Pune, India

[2]Professor (Electronics Engg Dept) S.K.N. College of Engineering, Vadagaon Bk. , Pune, India

[3]Asso. Professor (Computer Engineering) -- JJ MAGDUM College of Engineering , Jaysingpur, Dist.-- Kolhapur , India

ABSTRACT 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.

----------♦----------

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 [6]. 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 Policy Maker framework of Matt Blaze[5]. 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 [26]. 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. Attacks against these poor trust metrics dot the literature [31], and have been used to argue (incorrectly) that a centralized identity service is needed[13].

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 8 presents this design and discusses some open issues.

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.

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 with respect to attacks. 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 sub-graph 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."

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[5], 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.

Most trust metrics in the literature assume monotonicity. More precisely, if a monotonic trust metric accepts a node t as trustworthy when given a graph G, it will also accept node t when given a graph G0 that contains all trust edges in G. A primary motivation for this assumption is scaling. In particular, it allows for highly efficient overall distributed architectures in which all participants need only access a small, local subset of the global trust graph. Further, under such an assumption, there is no special vulnerability to denial of service attacks. With the monotonicity assumption, if an attacker can withold trust edges (i.e. cause the verifier to evaluate a subset of the global trust graph), then it cannot cause nodes to be accepted other than those which would be accepted given the entire trust graph.

The simplest trust metric is clearly monotonic. Adding edges to the trust graph can only cause more nodes to be reachable. 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[3]. 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 [31] 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[34], 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, this system does not quite meet the definition of "trust metric".

Reiter and Stubblebine[30] 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 [21]. 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. In addition, we present an analysis below (Section 2.6) suggesting that the Maurer metric is vulnerable to an easily-mounted attack. To do so, we will first need to prepare a framework for quantitatively evaluating the attack resistance of a trust metric.

Framework for analysis - What does it mean for a trust metric to be attack resistant? In this section, we present a quantitative framework for this question. In any given attack, we assume that the attacker can add or delete edges from the legitimate part of the trust graph, pointing to arbitrary nodes. These edges may point directly to nodes under the attacker's control, or perhaps to other good nodes, in order to fool the trust metric.

We assign a cost to each such attack. A typical cost metric is to count the number of edges added. Assuming an attack of a given cost, what is the highest number of bad assertions the attacker can force to be accepted? If this number is limited, the trust metric is attack resistant. If it can grow to the same order as the number of good assertions even for a fairly low cost attack, then the trust metric suffers from catastrophic failure and is not attack resistant.

Cost metrics: node vs edge attacks - We consider two cost metrics. The simplest is to count the number of edges added. Another useful metric is to count the number of "attacked " nodes with added outedges, and to assume that any such node may have an arbitrary number of edges added. As we will see later, the attack-resistance properties of different trust metrics behave differently under these two cost assumptions. Note that, for any given attack, the number of nodes counted is no greater than the number of edges counted.

An edge attack corresponds to fooling a victim into generating an edge. In many cases, mounting such an attack is straightforward. For example, the attacker may send a forged email pretending to be a friend of the victim, asking for a cert. Many such electronic means of transmitting key material suffer from lack of authentication.

Similarly, a node attack corresponds to taking over the victim's ability to create certificates. Such an attack is generally harder than an edge attack, but still feasible. For example, anyone with physical access to the victim's machine can recover the private key used to sign certificates, for example by using a keyboard sniffer to recover the encryption key used to protect the private key.

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, in degree of target node. We analyze two classes of attacks: one in which the attacker is able to choose the victims, and another in which the victims are chosen randomly. The former class of attack is at least as effective as the latter, and, not surprisingly, we find that it in most cases it is considerably more so. This result parallels the literature in scale-free networks, in which removing the "hubs" in a scale-free network with hub-and-spoke topology is far more effective in fragmenting the network than simply removing random nodes.

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 n2 nodes are attacked, entire system fails. Choose target as above, then spoof inedges of predecessors of target.

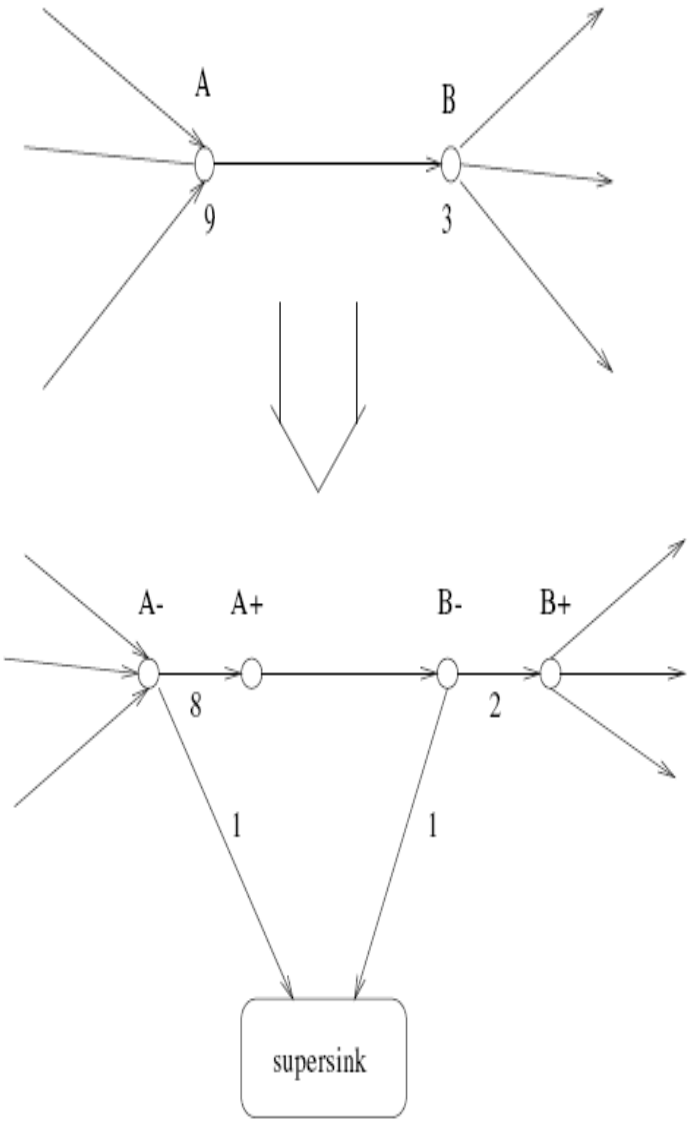
3. GROUP TRUST METRICS

This chapter presents the concept of group trust metrics, which have significantly better attack resistance properties than trust metrics under the scalar assumption. The key feature of a group trust metric is that it calculates a trust value for all the nodes in the graph at once, rather than calculating independently the trust value independently for each node. Thus, a successful attack causing one bad node to be trusted need not necessarily result in catastrophic compromise of the trust metric. Indeed, in this chapter we will present a simple trust metric, also based on max-flow over the trust graph, which has significantly better attack resistance than the theoretical best-case for scalar trust metrics.

To achieve attack-resistance, a group trust metric must sacrifice the monotonicity property - if the algorithm accepts a set of nodes S for graph G, then for a graph consisting of G plus additional edges, the set of nodes accepted may not be a superset of S. Otherwise, the attack mentioned in Section?? would be feasible.

However, for the trust metric presented below, a weaker monotonicity property does hold. As edges are added to the graph, the number of nodes accepted increases monotonically. As we will show in Section ??, this weaker monotonicity property fits some applications well, particularly those based on finding a majority or other consensus from the nodes accepted by the trust metric.

The Advogato network-flow trust metric - Capacity constrained flow network. Capacities of nodes are set as a function of distance from seed.



Comparison with Flake's Self-Organization work - The Advogato trust metric is quite similar to the Exact-Flow-Commnuity algorithm designed to discover communities existing on the Web[15]. Similarities include infinite flow to the seeds, finite, bounded capacities for the intermediate graph, unit capacity edges from all nodes to a supersink, and reaping of flow through these edges after a max-flow computation to determine membership in a community. The differences are worth noting:

- Exact-Flow-Community makes all edges bidirectional.
- Advogato places capacity constraints on nodes, while Exact-Flow-Community places capacity constraints on edges.
- Advogato uses distance from the seed to assign capacities, while Exact-Flow-Community uses a single constant capacity k for all intermediate graph edges.

The goals of Exact-Flow-Community and Advogato are somewhat different. The former is intended primarily to identify existing communities, based on patterns in Web links. Advogato also defines a community, but with the specific goal of attack resistance, in other words ensuring that an attacker cannot add an arbitrary number of false nodes to the true community.

For this goal, preserving the directionality of edges is crucial. Edges from bad nodes to good may be under the attacker's control, but edges from good nodes to bad are assumed not to be. Failure to distinguish between the two clearly leads to a loss of attack resistance.

Similarly, as described in Section 2.3, constraining node capacities yields better attack resistance than constraining edge capacities.

The relative significance of the two capacity assignment strategies is harder to determine. Advogato uses capacities dependent on the distance from the seed to accept a large number of nodes, even when the number of seed nodes (and their total outdegree) is small. The Exact-Flow-Community algorithm does not have this property. Thus, the authors propose an Approximate-Flow-Community algorithm that heuristically adds more seeds, then computes Exact-Flow-Community over this augmented seed set. It is quite plausible the results are similar to Advogato's, but we have not tested this hypothesis.

4. AN ATTACK-RESISTANT NAME SERVICE

Related work - There are many designs and implementations of systems to bind keys to names. The most popular is the X.509 family of standards, loosely synonymouse with the Public Key Infrastructure (PKI). In this framework, authority for the namespace is assigned to one or more certification authorities (CA's), generally composed of root CA's that delegate authority over some or all of the namespace to various subsidiary CA's. A consequence of this design decision is the root vulnerability—the compromise of any of the root CA's leads to catastrophic security failure.

Root vulnerability is a serious concern. Even though individual CA's may be well managed (with private keys

reasonably well protected), modern applications depending on PKI such as Web browsers ship with dozens of root CA keys enabled. Further, the grouwing popularity of CA software for off-the-shelf platforms (such as Entrust/PKI and Entegrity's Notary) implies that CA's will be deployed in contexts where it is difficult or impossible to protect the CA with a high degree of assurance.

Often, the CA given root authority is implementing a fairly simple and well defined policy. For example, VeriSign's Class 1 certificates are issued on a first-come, first-served basis. Domain name registrars issue second level domains on an effectively first-come, first-served basis, with the possibility of revoking the name (and issuing it to someone else) for non-payment of the bill, or as the result of a dispute. Actual implementation of these policies is at the whim of whoever holds the CA's private key. An attacker who succeeds in compromising the CA key can reassign names completely at will, with no regard to any policy.

It is possible to implement well defined policies and avoid root compromise. The naming service presented in this chapter factors the name service problem into to sub-problems: a policy language for expressing policies formally, and a distributed trusted third party for implementing the policies. In cases where the complex blend of factors controlling ownership of names can be distilled into a formal policy, our naming service can provide much higher assurance with lower certification cost than existing PKI's.

A design factored in this way can only be successful if two goals are met: the policy language should be rich enough to express a range of policies useful in the real world, and the distributed trusted third party should be trustworthy. Section 5.2 presents such a policy language, and Section ?? presents a distributed network based on an attack-resistant trust metric.

The policy language introduces the concept of asymmetry between initial registration of a name and updates to that name. In traditional PKI designs, the authority granted to the CA is total, so the CA has equal power to register and update name/key bindings. However, there are many interesting asymmetrical policies where the power to update is more restricted than the power to register. The first-come, first-served policy is an extreme example.

Authority to register new names is trivially granted, but authority to update existing names is never granted.

Implementation of naming service - In order to avoid a single point of failure, we choose a peer-to-peer network architecture for the implementation of the naming service. For simplicitly, we choose a two-level architecture, with a relatively small number of "server" nodes, and most clients contacting the server nodes to retrieve names and certificates.

All servers have knowledge of all other servers. Thus, notification of server joins and leaves are broadcast operations. When the number of servers is on the order of the square root of the total number of peers, network traffic is optimized. This square-root behavior is neither as bad as flat broadcast networks, such as the original Gnutella protocol, nor as good as the logarithmic performance of systems such as Chord[33].

Not all nodes store records for all names registered in the system. Rather, there is a "responsible server" relation, generally based on a hash function so that the set of responsible servers for a given name is effectively random. An average of 10 to 20 servers should be responsible for each name. This average is a tunable parameter. As it increases, security improves, but overall network traffic also increases.

There are essentially two different authorization decisions in the system. The first, evaluated by nodes when processing registration and update requests, is whether the request is valid. The second, evaluated by clients when performing queries, is whether the node responding to the query is trustworthy to provide the correct answer. Our design mirrors the fundamental asymmetry of these two decisions. Essentially, for mutation requests, clients send the request to all responsible nodes for the name, regardless of whether such nodes are to be considered trustworthy.

However, for query requests, clients only consider responses from nodes which are both responsible for the name and considered trustworthy.

Our design mixes up this clear separation a bit, because nodes processing mutation requests also function in the client role when they retrieve the policy and policy constraint associated with the parent name.

5. MESSAGE FLOW NETWORKS

The network flow-based trust metrics presented in previous chapters measure the flow of an abstract quantity through a trust graph, and use the presence of sufficient flow to a target node to decide whether that node should be accepted. In this chapter, we consider flow networks in which messages themselves are the unit of flow. Such networks are especially useful for spam-resistant messaging. Indeed, using a trust metric for access control is vulnerable to attacks involving huge volumes of spam messages. The goal of a message flow network is to tightly

bound the volume of spam messages which may be succesfully delivered by bad nodes.

The basic message flow design –

Assumptions - We assume a trust network with similar properties as in previous chapters. Each node in the network represents a participant in the overall system. Each user manually enters trust edges into the graph. Each edge is annotated with a message volume associated with the peer.

If a user s wants to receive no more than k messages per time unit from peer t, then s enters a trust edge pointing from t to s, annotated with k units of flow. Note that the direction of this trust edge is reversed with respect to the trust metric presented in Chapter 3. We also assume a central server which knows the entire trust graph, and also keeps track of dynamic information based on the actual flow of messages sent through the system.

Message flow - In particular, the server maintains a residual capacity for each trust edge in the network, named in analogy to the standard algorithm for computing maximum network flows. However, instead of attempting to saturate the graph with the maximal number of augmenting paths, residual capacity is consumed only when messages are actually sent.

In more detail, when a node x wishes to send a message to node s, the server attempts to find a path from x to s through the residual capacity graph. If no such path exists (in other words, if there exists a partition of the graph with x on one side and s on the other in which all trust edges have zero residual capacity), then the message is blocked. Otherwise, the message is sent, and the residual capacity for each edge along the path is decremented by one.

As usual in computing augmenting paths, it's a good idea to use the heuristic of choosing a shortest path through the residual graph, thereby minimizing total capacity consumed for an individual message.

Repleneshing the flow - Edge capacity is measured in units of number of messages per unit of time. In addition to consuming capacity when messages are sent, there must also be a mechanism for replenishing this capacity, so that the network is roughly in equilibrium when there is a steady stream of messages.

There are a number of approaches to this repleneshing. Perhaps the simplest is to replenish all residual capacities to match the maximum capacity specified in the trust graph, once every time unit. However, this approach has a number of disadvantages, including an uneven probability of rejecting messages over time, depending on the fractional time since the last increment of the time unit.

It would also be desirable to accommodate bursty traffic patterns. If a user specifies seven messages per week from another peer, it would be unusual to expect messages sent at intervals of exactly 24 hours. Rather, it's more likely that several days would pass without a message, interspersed with bursts of several messages in quick succession.

Thus, we propose a computationally simple and efficient method modelling a process by which flow is replenished with exponential decay. In addition to storing a scalar for each edge representing the residual capacity, the server also stores a timestamp representing the last moment that the capacity was updated.

Then, each time the capacity is queried or modified, it is updated according to the following simple algorithm:

$$cap = cap + (max - cap) \cdot (1 - \exp(timestamp - now))$$
$$timestamp = now$$

It should be clear that, over a period of one time unit, the total flow across such an edge will never exceed the maximum specified. At the same time, burstiness is penalized, but only somewhat. For example, if messages are sent in bursts once every time unit, interspersed by inactivity, then the total flow is $1 - e^{-1} \approx .632$ of the maximum specified.

Security Analysis - Outline: partition network into good & bad.

Total message flow across partition is bounded by total capacity of edges across the partition. Consider two cases of partition:

1. There is one bad node. Then he can't send more spam than the total cap of edges from good nodes linking in.

2. There is one good node. Then you can't receive more spam than the total cap of edges to you. Have a heuristic that favors short paths when multiple messages are competing for scarce flow. Then, hopefully all the good mail from nearby neighbors gets through, and at worst the spammer blocks only mail from good peers farther away.

Pragmatics - The design of this chapter is unimplemented as of this writing. However, it should be straightforward to implement as a centralized web service.

Obviously, having a centralized server is limits both scaling and security (all users are vulnerable to an attack on the central server). A completely decentralized peer-to-peer network implementing the core concepts of the message flow network presented above is the topic of the next chapter.

Application Notes - Outline: email is killer app for this. A similar app is posting comments and backlinks in blogs. Since # nodes is so much smaller, a centralized server may be appropriate. Don't need to actually send messages through net. Don't need to replace SMTP. You can just send tokens authorizing the sending of a message. Then, the sending email client can query the server for such a token, and insert it into the headers of the mail. The recepient email client extracts the token from the headers, and verifies it.

6. CONCLUSION

Several conclusions follow from the analysis presented in preceding sections. First, the scalar trust metrics place a very low upper bound on the attack resistance possible to achieve. Second, we demonstrate that simple trust metrics based on maximum network flow meet these upper bounds. Thus, within these assumptions, there is little if any room to design improved trust metrics.

In particular, analysis of Maurer's trust metric shows its attack resistance to be both disappointing and equivalent to much simpler metrics.

A reasonable conclusion, then, is that it is the monotonicity assumption itself which is flawed, and that it is not possible to achieve good attack resistance by verifying a small, local subset of the trust edges comprising the global trust metric. Indeed, in the next chapter, we show that by relaxing the monotonicity assumption, a reasonably simple trust metric also based on maximum network flows can acheive dramatically better attack resistance.

BIBLIOGRAPHY

1. Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. First Monday, September 2000.

2. Albert-La´azl´o, R´eka Albert, and Hawoong Jeong. Scale-free characteristics of random netwoks: The topology of the world wide web. submitted to Elsevier Preprint, August 1999.

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

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

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

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

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

8. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash (extended abstract), 1989.

9. 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.

10. 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.

11. Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In Workshop on Design Issues in Anonymity and Unobservability, number 2009 in LNCS, pages 67–95, 2000.

12. Roger Dingledine and Paul Syverson. Reliable MIX cascade networks through reputation. In Proc. Financial Cryptography, March 2002.

13. John R. Douceur. The sybil attack. In Proc. 1st International Workshop on Peer-to-Peer Systems, March 2002.

14. Cynthia Dwork and Moni Naor. Pricing via processing or combating junk mail. In Ernest F. Brickell, editor, Advances in Cryptology – CRYPTO '92, number 740 in LNCS, pages 139–147. Springer-Verlag, 1992.

15. 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.

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

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

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

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

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

21. 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.

22. Petar Maymounkov and David Mazi`eres. Kademlia: A peer-to-peer information system based on the XOR metric. In Proc. 1st International Workshop on Peer-to-Peer Systems, March 2002.

23. Mark S. Miller and K. Eric Drexler. Incentive engineering: for computational resource management. In Bernardo Huberman, editor, The Ecology of Computation, pages 231–266. Elsevier Science Publishers/North-Holland, 1988.

24. Mark S. Miller and K. Eric Drexler. Markets and computation: Agoric open systems. In Bernardo Huberman, editor, The Ecology of Computation, pages 133–176. Elsevier Science Publishers/North-Holland, 1988.

25. Tim Moreton and Andrew Twigg. Trading in trust, tokens and stamps. In Proc. Workshop on Economics of Peer-to-Peer Systems, June 2003.

26. Andy Oram, editor. Peer to Peer. O'Reilly & Associates, 2001.

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

28. David Post, 1999.

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

30. 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.

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

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

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

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

35. Mads Tofte and Jean-Pierre Talpin. Region-based memory management. Information and Computation, 1997.

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