An Architecture for Developing Behavioral History

Mark Allman[†], Ethan Blanton[‡], Vern Paxson[†]
[†]*International Computer Science Institute,* [‡]*Purdue University*

Abstract– We present an architecture for large-scale sharing of past behavioral patterns about network actors (e.g., hosts or email addresses) in an effort to inform policy decisions about how to treat future interactions. In our system, entities can submit reports of certain observed behavior (particularly attacks) to a distributed database. When deciding whether to provide services to a given actor, users can then consult the database to obtain a global history of the actor's past activity. Three key elements of our system are: *(i)* we do not require a hard-and-fast notion of *identity*, *(ii)* we presume that users make *local* decisions regarding the *reputations* developed by the contributors to the system as the basis of the trust to place in the information, *(iii)* we envision enabling *witnesses* to attest that certain activity was observed *without* requiring the witness to agree as to the behavioral meaning of the activity. We sketch an architecture for such a system that we believe the community could benefit from and collectively build.

1 Introduction

A key problem in trying to prevent unwanted traffic is that interactions on the Internet are largely anonymous and self-contained. The host offering services often has no idea who is requesting the service or what sorts of activity the requester has undertaken in the recent past. Lightweight anonymous access to information has been fundamental to the proliferation of network services. However, the Internet's wide-open nature has also fueled the spread of worms, viruses, spam, DDoS attacks, and other forms of unwanted traffic.

Consider some requester, R, attempting to access some service, S. What does S know about R? Several bits of information *may* be available, such as:

- If IPsec [7] or TCP's MD5 option [5] are used then S may have a reasonably solid notion of the *identity* of R, and therefore can make a determination of whether to provide access to the given service.
- S may be able to query a local cache of R's previous activity. For instance, a local Intrusion Detection System (IDS) may track behavior exhibited by remote hosts over time.
- S may have access to information about R collected and shared by remote entities. For instance, [11] outlines a scheme for one instance of Bro to receive event notifications over the network from other Bro's. Additionally, systems such as *www.dshield.org* aggregate simple activity reports from a vast number of locations to form a picture of a host's recent activity. Another class of mechanisms proactively test hosts to determine some characteristic, which is then publicly disclosed (e.g., R is an open SMTP relay).
- S could check R against a list of host "types". For instance, lists exist that identify DSL, cable modem or dial-up IP addresses (e.g., *www.sorbs.net*), and a number of spam-fighting systems block or score email based on whether a remote host is on such a list.

The above systems all provide useful information that can help form policy. However, the information is quite often narrow in scope and/or difficult to obtain — especially when trying to thwart unwanted traffic in real-time. For instance, locally-collected IDS information may be quite rich, but it only relates to hosts that attempt to access the local network. Further, setting up pair-wise trust between IDS systems is an arduous process when done at any scale. Likewise, setting up and managing pair-wise keys for IPsec or MD5 can be burdensome and contrary to the nature of providing publicly available services. Finally, databases containing the "type" of IP address offer a broad view of the network, but offer no indication of the behavior of individual hosts.

Our goal is to overcome these limitations. We envision doing so by devising a system that can accumulate reports of unwanted traffic in a general fashion across the entire network. As mentioned above, there are some systems currently providing instances of such services. However, given their centralized nature these lack the robustness we would desire in a system called upon to absorb large numbers of queries and provide actionable information that we can use in real-time. In addition, the consumers of the data provided by these services must both trust the information and agree with the policies used to aggregate the reports and derive the information. Finally, we believe an opportunity exists to consider a *general* architecture for tracking the behavior of large numbers of actors, where both "behavior" and "actor"

can encompass a wide range (as discussed below).

In this paper we elucidate an architecture for a *distributed* system that provides a lightweight actor-based history database. Rather than requiring a large series of pair-wise key exchanges, a user of the database makes *local* decisions regarding the degree to which to trust information in the database, primarily in terms of the user's local assessment of the submitter's *reputation*. While we primarily illustrate the architecture in terms of tracking purported attacks associated with Internet hosts, the architecture generalizes to tracking general types of purported *behavior* (e.g., attacks) associated with *actors* (e.g., Internet hosts or IP addresses).

The system we propose does not require aggregation of information in the database into a succinct "status" of a given actor. Rather, the database provides information to consumers who make their own determination as to the validity and trustworthiness of the information, and then formulate a local policy decision based on these assessments.

By having judgments of reputations and reports remain local decisions, we aim to address some of the issues with how users can possibly trust such a wide-open database that will inevitably include misinformation. An additional, and arguably more powerful, mechanism for developing such trust is the inclusion in the system of *witness* reports: these are additional records that corroborate part of the *context* of other reports. For example, a router might corroborate that a certain packet passed its way, without needing to agree that the packet constituted an attack. Because the router is attesting to a low-level fact rather than a high-level judgment, the hope is that well-known parties (e.g., ISPs) will find it affordable in terms of both processing and reputation to contribute such reports.

This paper presents an architectural overview of the system, not a detailed analysis of any particular facet. Much more work must be done by the community to bring such a system to fruition. Our hope is to refine the thinking behind the system by receiving community input. We present an overview of the information in the database in § 2; discuss the role of policy in § 3; analyze attacks on the system in § 4; present additional issues in § 5; and sketch future work in § 6.

2 Database Overview

We maintain the history database in a Distributed Hash Table (DHT) [3] designed to support querying in massively distributed environments (e.g., PIER [6]). DHTs provide a robust and distributed system for storing information, meaning that each DHT node need only cache a fraction of the information in the database, while complete information is efficiently made available to every node. The use of a DHT also minimizes the amount of damage that duplicitous database nodes can cause, while encouraging arbitrary systems to join the database.

Entries in the database are cryptographically signed by the entity which submits them. Each record is inserted under two hash keys: one hash key identifies the actor about whom the entry relates, and the other hash key is a cryptographic public key associated with the submitter. These allow consumers to look up behavior associated with particular actors (e.g., the IP address of a remote host attempting to access one of the consumer's services) and the set of all records submitted under a given key, respectively.

The cryptographic keys used in the system need not be widely known or authenticated in any way; they serve only to correlate records in the database by tying them to a particular submitter. For our purposes, it is often enough to know that two records were submitted by the same entity without necessarily having a clear idea of who that entity is. The use of cryptographic signatures allows entities which retrieve a set of records to have some degree of confidence that the records were inserted by the same entity (or at least inserted under collusion), as well as opening the possibility for local policy decisions to "trust" certain foreign keys. We discuss this and other opportunities for using keys to influence policy in § 3.

We store several kinds of information in the hash table. The basic record is a *behavior record*: a statement that the reporting entity believes it observed a given type of behavior perpetrated by a given actor (e.g., that a given IP address launched a given attack). Records may also be inserted by *witnesses* that vouch that certain events occurred, but not whether these events reflected the purported behavior. For example, a witness could report that they observed a particular packet, without needing to agree that it constituted an attack. Such records form *audit trails* that can lend credibility to reports of particular behaviors. Finally, entities which make use of a particular record in the database may note that they have done so, and thus make it known that they have some level of trust in the record. Or they might note that they declined to make use of a particular record. In either case, they might later record their post facto assessment of their own decision.

2.1 Behavior: Attacks

In this section we illustrate the general framework for tracking behavior using attacks from Internet hosts as a specific example. The reader should however keep in mind that the scope of "actor" and "behavior" could in principle be considerably broader. For example, the domain of actors might be email From addresses, with the corresponding behaviors reflecting the use of the addresses in phishing or spamming email. As an example that focuses on behavior but not attacks, we might

imagine a system for which actors are blogger Web page URLs, with the associated reports being submitters' assessments of the accuracy and utility of the postings.

For tracking attack behavior, when a participating host or edge network (e.g., an IDS) determines that "unwanted" traffic has arrived they may indicate this by inserting a corresponding record in the distributed database. They would insert the record twice, once using the IP address (i.e., actor's identity) of the purported malicious host as the hash key and once with the reporter's public key as the hash key. Each record might consist of the following fields:

- *Timestamp*: The time when the first instance of the behavior was received. The DHT node that stores the record also records a *report timestamp* indicating when it received the report. As discussed in § 4, the relative time reports are made could have implications in terms of the reputation of the reporter (and, therefore, independent timestamps may be useful).
- *Actor identity*: A characteristic of the actor creating the behavior that can be used to retrieve reports of that actor's behavior. For some types of behavior, the identity to associate with them may appear obvious. For example, for observations of network attacks, the IP address of the purportedly malicious host will often make sense. Clearly, equating identity with an IP address is not quite right. However, IP addresses can be acted upon in enforcing security policy. Also, future namespaces (e.g., HIP [9]) could be readily incorporated into the architecture.

  In other cases, the notion might be more diffuse, and might even involve *multiple* identities. For example, testing email for possible spam might be done in terms of looking up each of the relaying servers listed in "Received:" headers. Thus, we may find it useful to think in terms of "identity components" or "identity hints", rather than a single notion of identity.
- *Protocol* and *port number*: The reporter could optionally report the protocol and port numbers on which the unwanted traffic arrives. In some cases this could be useful, for example when subsequent traffic from the given host is likely benign (e.g., web requests from a host with an open SMTP relay are not necessarily likely to be problematic). In other cases, an attack may span a wide variety of ports and/or protocols (e.g., a port scan), in which case this information is difficult to express and has little utility.
- *Behavior observed*: a code roughly describing the unwanted traffic observed. We envision encoding the behavior in broad categories rather than trying to highlight specifics. For instance, a reporter of attacks would indicate "worm", not "Code Red". Possible attack categories might be: worm, virus, spam, scanner, DDoS attack, etc.
- *Behavior digest*: This field contains a fingerprint of the specific activity observed. For an attack, it might be a packet digest [10] of the packet that triggered the report. The point of capturing a behavior digest is to correlate the record with audit trails (as discussed below).
- *Signature*: the cryptographic signature of the above fields and the associated public key.

Defining when to place something in the database is behavior-dependent. For instance, if an IDS finds the signature for a well known attack (e.g., Slammer) then it could record an event immediately. However, an *SSH* connection that attempts to log in with a bad username one time may not be cause for recording an event (given that someone might have made a mistake or a typo). Or, this might warrant a delay in the inserting of the event in the database to better ascertain whether the *SSH* service is under attack.

2.2 Witnesses

To add credence to a reporter's database entry we allow for the insertion of *witness statements* in the database. These records do not indicate particular behavior, rather they indicate that the witness indeed observed some element that is purported to be part of the behavior. One possible source of witnesses would be in the routers that forwarded the unwanted traffic to the reporter. If such routers kept records of the traffic traversing their networks in digest form (e.g., for traceback [10] or for reporting packet obituaries [2]), the reporter could request these systems to insert witness statements into the database (signed by the witness' public key).

The statements form the basis of an *audit trail* that at least supports the reporter's contention that a particular traffic stream arrived. In keeping with the locally-determined nature of decision-making in our system, the precise use of this audit trail in assessing a reporter's reputation or setting policy is left for each database user to determine.

2.3 Signatories

A final item for the database to store is annotations of records based on their use (or not) in setting policy. In its simplest form, sites or hosts that make use of a record in forming a decision to treat some traffic in a non-standard way (blocking, rate-limiting, heightened surveillance) can sign that record to indicate that they used the information to form policy. These hosts are indicating that they have used this record as part of their policy in how to treat an actor without actually observing specific behavior (which would cause an independent behavior report

to be entered). The signatory mechanism is designed to play a part in determining the reputation of reporters by recording which of these reports other sites have found useful.

The decision to use a particular attack record in the database, or to sign and report its usage, does not need to be solely based on the information from the database. It could be partially based on local policy and observation, as well. For instance, a report of an IP address as a scanner by some arbitrary reporter may not be believed on its own. However, if an incoming packet from that address is destined for a dark portion of a site's address space, then these two pieces of information together may be enough to enact a policy of blocking the given IP address. In this case, the given record in the database should be signed to indicate that it played a part in designing a particular policy.

Finally, we envision two possible refinements to the use of signatories. The first is the reporting of negative use; i.e., that a site decided not to use a reporter's report about a given actor. This allows other sites to more quickly learn of the skepticism of peers they deem trustworthy. The second is *follow-up*, in which sites later report whether subsequent activity led them to a different conclusion than they formed initially. Follow-up is basically a form of report revocation, per § 5.

3 Policy

The point of the database is to inform local security decisions with a broader context than that which can be locally collected. The database does not determine that a given actor is somehow malicious (or, more generally, exhibited a particular behavior). Rather, it simply stores and provides reports from sites that have made such determinations. The reports from the distributed database can be combined with any other information on-hand (e.g., IDS records or topology information) to make policy decisions. In using such reports a key component is *trust*. The consumers of the reports have to somehow trust that the report providers inserted valid records in the database.

Database records could be invalid for three reasons: (i) the record provider makes inaccurate assessments of an actor's behavior, (ii) the record provider is intentionally inserting inaccurate records in an attempt to use the database as the basis of a denial-of-service attack on an actor (or, more generally, to *frame* an actor), or (iii) the information contained in the record was accurate at one time but is now out-of-date. To address the first two validity issues we depend on a reporter's *locally-determined reputation* to assess the degree to which the information will be trusted. The third issue is addressed by aging information in the database.

As discussed in § 2, each entry in the database is signed by the reporter. We do not impose a tight coupling between the identity of the reporter and the key used to sign records; rather the history of the key's reports can be used to assess a reporter's aptitude in determining malicious behavior. Reputation is a *local* calculation based on various pieces of information culled from the database. See, for example, [1] for an illustration of one particular scheme for using a distributed database to assess the reputation of participants in a peer-to-peer system. We leave a concrete definition of reputation as future work (and, indeed, there can be various calculations depending on what a particular site values); however, we note several aspects that could be taken into account when assessing the fidelity of reporter X:

- The number of distinct reporters (and their reputations) making the same assessment as X. This lends weight to the fact that X is not inaccurate or malicious.
- The number of signatories on reports made by X. This indicates how many others have some amount of trust in the given report (perhaps due to corroborating evidence, as discussed in § 2).
- An audit trail of the events that X has reported. While the third-party statements that form the audit trail do not confirm X's assessment of some property of the given actor, they do offer evidence that X did in fact observe a traffic stream with the given digest (with a high probability). This can be used as evidence that X may be acting honestly.
- Local evidence that concurs with X's findings; for instance, if X identifies some host H as a scanner and local information has not yet made that determination but has noted several attempts by H to access dark IP addresses.
- Reputations can be earned within different contexts. For instance, a particular X may have a highly accurate worm detector, while having a sub-par scan detector that is often wrong.

Each of the above bases for reputation can be gamed by a malicious reporter. The algorithms for deriving a reporter's reputation therefore need to be tuned to be skeptical of a reporter until the reporter has a track record that is supported by evidence and previously well-known reporters. See § 5 regarding the problem of bootstrapping such a system.

While assessment of reputation can be used to take care of some aspects of invalid information in the database, it cannot account for out-dated information. In fact, out-dated information may well lead to the reputation of a reporter suffering because a once-valid report is now seen as invalid. Therefore, we need some way to age out old information. If we retain the information in the database and allow users to apply their own aging poli-

cies, this would likely yield a database full of information that consumers always essentially factor out, wasting resources. Therefore, we likely need the DHT nodes to prune "old" information, either deleting it or perhaps using sketch-like schemes [8] to aggregate older information in a form of graceful degradation.

4 Cheating

A key component of the system sketched above is that it does not require the identity of the actors to be known, but rather depends on the track record of various entities. This leaves the door open for abuse from malicious actors. One concern in this regard is misbehavior by the nodes participating in the DHT, to prevent legitimate information from being inserted into the database or from reaching users in response to queries. In general, DHTs are susceptible to a variety of attacks [4], and resisting these (for example, by restricting membership in the DHT) is an area of active research that is not unique to our needs. Thus, in this section we focus on a second class of cheating, namely attackers inserting bogus information in the database that is, itself, an attack (e.g., noting that one's enemy has a worm in the hopes that this will cause denial of service to said enemy).

In previous work, [1] shows that a workable reputation system can be built such that reporters providing a good amount of bogus information can be identified as cheaters. However, that work also shows that catching reporters that generate only a small amount of misinformation is difficult. Therefore, additional reputation assessment techniques are likely needed, based on the sorts of misinformation that could be placed into the database in our system. We consider the four kinds of misinformation an attacker could place into the database.

First, a malicious user could attempt to place bogus reports in the database in the hopes of denying service to the given hosts. For dealing with some simple versions of such attacks, see [1]. However, a more motivated attacker could retrieve database entries for a key, K, that represents a solid contributor (which could be determined by looking at signatories or by running some known, widely-used reputation algorithms) and then insert attack records with the same information as those submitted by K but signed with the attacker's key, K'. If the attacker monitors the database for a period of time and continues to report entries based on those of K (or, even a few keys to mask the blatant copying) then K' can steal solid reputation built by K, and at this point K' can start inserting bogus records.

One way to resist such attacks is to incorporate the time when an attack was reported, with the first reporter getting more "reputation points" than subsequent reporters. For this defense to work, it is critical that the database correctly record reporting times. In addition, the reputation assessment may highlight "unique reports" to ferret out reports that only K' makes that others do not (though this can be countered with multiple malicious keys colluding). More speculatively, an honest, high-reputation reporter could occasionally insert "ringers" into the database: false records that are revealed as such after a delay. Any other reporter who also reported the ringer is unmasked at that point as having copied the original report.

A second form of misinformation is hosts filing false witness statements in an attempt to bolster their false attack claims. This might be mitigated by asking ISPs to sign all their witness reports with well-known keys.

Third, hosts could add fake signatories to falsified attack reports in an effort to add weight to those reports. If the signer of some record engaged in the above "reputation stealing" scheme then it is conceivable that hosts could be fooled into impeding an innocent actor's traffic. The reputation assessment could take into account the set of signatories in an effort to see if they overlap with known solid keys, or not.

Finally, attackers could flood the database with useless records, to render legitimate use of the database computationally infeasible, to greatly complicate reputation calculations, or even for purposes of spamming advertisements if a mechanism exists that will display entries in the database to users (e.g., for error handling). This threat is quite significant because the system purposely does not tie reporter keys to specific entities. Therefore, it lacks the means to prevent an attacker from minting an endless stream of keys to avoid having a single key identified as a flooder.

It may be possible to diminish the problem of flooding by requiring that reports be corroborated by witnesses that themselves have established reputation in a more global manner (perhaps by independent auditing, as discussed below in § 5). Clearly, there is much future work to do in terms of developing reputation assessment techniques that can ferret out or resist complex patterns of misbehavior.

5 Other Issues

In this section we tackle a variety of topics that require additional work as the community moves from an architectural concept to building a real system.

Deployment: The architecture we sketch should be incrementally deployable because users do not require global coverage to gain benefit from the system. However, bootstrapping issues remain, both in terms of starting the system as a whole and then adding reporters later. What sort of bar must a reporter meet before they can be reasonably trusted? How does a new consumer of the information in the database know whom to trust? Is there some sort of calibration system that could be developed

to help new users of the database verify which reporters are providing solid information? The path for a new reporter seems less burdensome because the reporter is just informing the database of local decisions already being made. As the reporter develops a track record, the reported entries gain more usefulness globally. This does not greatly benefit the reporter, but perhaps no additional incentive is needed in this regard — it is annoying to be attacked, and for many victims satisfying by itself to engage in some sort of response, even one without direct benefit.

Linking Keys and Identity: A key to the scalability of the system is that keys are linked to reputations (which can be independently computed). However, there are pros and cons associated with keys becoming linked with identity. For instance, if network operator A told network operator B, whom they trusted, how to identify the reports A sent to the database, then B may decide to trust those reports without bothering to compute a reputation for A. (Of course, this can work in the reverse and B could decide that A is sloppy and thus untrustworthy.)

However, if an adversary knows that a particular key reports attacks on a given address space, then they could avoid that address space to avoid being reported. (E.g., it is easy enough to check the database after performing a scan to see if the attack triggered an entry, and if so, which key was used to sign it.) If attackers avoid networks that report malicious activity, this can create an incentive for network operators to report malicious activity in the database. On the other hand, in a global sense this could work against trying to quickly contain malicious hosts.

Additionally, if an attacker determines that some address space is linked to a particular public key in the database, the user could use this information in an attempt to determine the site's security policy. For instance, a malicious user may be able to determine what sort of criteria the site uses to declare that a host is a scanner. Such knowledge may let the user explore the site (from additional compromised hosts, say) without triggering alarms.

Finally, if a key is known to represent some organization and the organization's reports to the database are often wrong, then this could cause embarrassment to the organization (and, hence, be a disincentive for participating).

Revoking Reports: Another aspect of the system that requires thought is how (or whether) a given reporter corrects mistakes it has inserted into the database (by annotating, not deleting records). For instance, suppose someone from a given host attempted to *SSH* into some system using an invalid username. This is not an infrequent attack method, and as such it could be reported. However, if this was later determined to be a simple case of a user making a typo in a username, then it seems reasonable for the reporter to indicate that the given report was benign after all. Allowing the database to support this is not difficult, but the question remains of what happens within the reputation assessment. The assessment should probably give the reporter some credit for correcting a mistake. However, at the same time we could view the need to correct as an indication of sloppiness because the maliciousness was not soundly assessed before entering the initial report.

A more speculative application of revocations would be after an infected machine is cleaned up. For instance, if some host was flagged as spewing spam and later closed an open relay, one approach would be for the reporters that noted the spammer to indicate that they now believed the host was no longer sending spam. This path becomes messy because it requires that the reporters have some way to determine a given host has been cleaned up. Alternatively, the architecture could allow for a separate entity (e.g., the victim's ISP) to enter such a record (and of course they would accrue reputation for the fidelity of their cleanup reports).

Finally, the system needs to support key revocation in order to flag a previously-valid key as now compromised and untrustworthy, lest an attacker steal the key's reputation. This could be done by inserting a self-signed revocation in the database as one of the entries under the key's hash.

System Openness: The openness of the system is a balancing act. Ideally, we would like to allow only honest participants and to deny anyone with ill-intent. The degree to which the system can remain open may largely hinge on the power of the reputation assessment techniques used and the system's ability to resist database flooding attacks.

As discussed thus far, we have considered the system to be open for use by anyone. However, there are advantages to restricting membership in the DHT itself to only known parties to reduce the trustworthiness issues involved in the infrastructure that supports the database. Likewise, closing audit reports to a certain subset of participants also seems tractable and useful.

In addition, the entire system could be instantiated multiple times, each scoped in terms of who can report to and/or query from the database. Such a scoping could prove burdensome due to additional credential requirement. On the other hand, the credentials would be known by the DHT and would still likely be easier to manage than the pair-wise interactions currently required for sites to share information. Furthermore, the complexity of the reputation assessment system in such an environment, while not being completely eliminated, would likely be substantially reduced (e.g., because the likelihood of "reputation stealing" attacks would be quite

small). Finally, we note that for some environments, such as the Grid, our system could leverage an existing identity infrastructure.

Overhead: The system outlined in this paper has a cost in terms of the network resources used. The worst case is that each session establishment initiated by a remote host would require a database query to assess the possible intent of the requester. Even a modest-sized institute such as ICSI (a few hundred hosts and a dozen public servers) is visited by many thousands of remote hosts each day, and this figure could become much larger if an attacker were spoofing IP source addresses, though we could offset this by requiring that connections first establish before then deferring their further progress until we can vet their source. We could also gain benefit by caching the lookups from the database, but must do so carefully to avoid using stale information in the face of a fast-spreading attack.

Surrogates: A final avenue for investigation is the use of *surrogates* to aggregate the information from the database such that local resources are not required for this task. Clearly, this requires that a consumer trust the surrogate to accurately aggregate information from the database in a timely manner; in particular, to precompute reputations of reporters (as opposed to summarizing per-actor behavior). Such trust could be established based on external relationships. For instance, a company's field offices could trust a surrogate at headquarters to do the computational work of determining reporter reputations nightly and then distributing the information. More speculatively public web sites could be setup to serve information about reputations (or even their judgment of given actors) based on the information culled from the database. In this case, trust goes back to the track record of the surrogate's reports. Also, given that the information being aggregated is generally available, we can audit aggregation sites periodically to verify their trustworthiness.

As a practical matter it may be worthwhile to store snapshots of the database periodically in "data warehouses". Whereas our belief is that information about the distant past is not useful in terms of current malicious activity, such a warehouse would allow us to test this hypothesis. In addition, new reputation algorithms and circumvention attacks based on the database's timeouts can be analyzed.

6 Future Work

We have sketched the beginnings of an architecture to store and retrieve activity reports. Here we focussed on the big picture, but to bring such a system to fruition will require community efforts on a number of fronts: (i) developing workable locally-computable reputation algorithms, (ii) obtaining acceptable performance in terms of lookup overhead, timeliness of information propagation, and resilience to flooding, and (iii) devising workable witness digests and associated audit trails, as a key element in providing a low barrier-to-entry when developing a reputation.

Acknowledgments

Many thanks to Joe Hellerstein, Petros Maniatis, Scott Shenker and the anonymous reviewers for their thoughtful comments. This work was supported in part by the National Science Foundation under grants ITR/ANI-0205519, NSF-0433702 and STI-0334088, for which we are grateful.

References

[1] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, 2001.

[2] K. Argyraki, P. Maniatis, D. Cheriton, and S. Shenker. Providing Packet Obituaries. In *Proceedings of ACM SIGCOMM HotNets-III*, 2004.

[3] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking Up Data in P2P Systems. *Communications of the ACM*, 46(2):43–48, Feb. 2003.

[4] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Security for Peer-to-Peer Routing Overlays. In *Fifth Symposium on Operating Systems Design and Implementation (OSDI '02)*, Dec. 2002.

[5] A. Heffernan. Protection of BGP Sessions via the TCP MD5 Signature Option, Aug. 1998. RFC 2385.

[6] R. Huebsch, J. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *Proceedings of the* 29^{th} *VLDB Conference*, 2003.

[7] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol, Nov. 1998. RFC 2401.

[8] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based Change Detection: Methods, Evaluation, and Applications. In *ACM SIGCOMM Internet Measurement Conference*, Oct. 2003.

[9] R. Moskowitz and P. Nikander. Host Identity Protocol Architecture, Jan. 2004. Internet-Draft draft-ietf-hip-arch-02.txt (work in progress).

[10] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Single-IP Packet Traceback. *IEEE/ACM Transactions on Networking*, 10(6):721–734, Dec. 2002.

[11] R. Sommer and V. Paxson. Exploiting Independent State For Network Intrusion Detection. Technical Report TUM-I0420, Technische Universitat Munchen, Nov. 2004.