Reputation in Privacy Enhancing Technologies

Roger Dingledine, Nick Mathewson, and

Reputation Technologies, Inc.

Paul Syverson

Naval Research Laboratory

Reputation is the linchpin of a dynamic and pseudonymous future. In a networked world where individuals interact via anonymous remailers, and where the online services they use are themselves provided by an ever-changing pool of semi-anonymous users, the distinction between pseudonym and identity blurs. In this world, reputation is one of the few tools that can still provide trust — trust among the users of distributed services, and even the trust necessary to maintain reliability and accountability of these services.

In its most general form, reputation is memory about past performance. This memory can be localized and idiosyncratic, as in the case of users that remember which servers have worked well in the past; centralized and shared, as in the case of an auction site that tracks customer satisfaction of various vendors; distributed and shared, as in the case of servers that vote one another into different reliability categories; or even implicit within the structure of the system itself, as in the case of systems that embody trust as microcurrency that reliable systems tend to accumulate.

While reputation might superficially seem inimical to privacy concerns, systems with explicit reputation can actually *enable* privacy by controlling the flow of information about pseudonymous individuals, and reducing the demand for out-of-line information exposure.

As with security, it is tempting but incorrect to think that reputation is a simple matter of bolting an extra service to the side of an existing system. This point is illustrated by two reputation systems that have been designed for use in remailer networks.

An Example: Remailer Networks

Remailer networks allow people to send and receive mail while protecting their identities. Today's remailer networks use a handful of long-lived, static servers with fairly uniform reliability. Currently deployed reputation systems send periodic test messages through each remailer to determine which are

Copyright is held by the author/owner.

currently working. This "pinging" approach works well enough for a small static list of servers. However, a network that is made of a small static set of remailers is potentially vulnerable to a well-funded adversary in a variety of ways, e.g., denial of service or remailer compromise. Any solution based only on hardening the nodes runs contrary to the basic premise of remailers that trust is distributed rather than mutually assured. Thus, to better resist a well-funded adversary, the remailer network must grow so it has enough nodes to properly distribute trust. Pinging for reputation breaks down in an environment where the network is made up of many transient volunteer nodes. On the other hand, an adversary might render a growing, dynamic network useless by volunteering a flood of unreliable remailers, or it might manipulate the reputation system to improve the standing of remailers it owns.

The first design, presented in [1], describes a reputation system for improving remailer reliability. Remailer reputation is based on both positive and negative performance. It is not enough to just keep track of failures, because new unreliable remailers would be rated the same or better than remailers that consistently perform well. In this design, each remailer in the message's path passes back a receipt to the one behind it. Senders can successively query for receipts to determine which remailer to blame for delivery failure. However, a remailer might refuse to provide a receipt for a particular message either because it failed to send the message, or because it was unable to obtain a receipt from the next hop. We solve the problem of pinpointing failures by introducing a set of weakly trusted global witnesses. These witnesses are contacted when the next hop in the path refuses the message, allowing a remailer to prove that it made a best-effort delivery attempt. Senders can also tell witnesses about remailers that silently dropped messages (meaning they got a copy but did not attempt to pass it on). These witnesses verify and tally failures, and also send their own test messages to distinguish reliable remailers from new ones that have not yet been tested. Reputations are tabulated and made available to client software, which can use them to choose reliable remailers for sending anonymous mail.

This reputation system attempts to improve reliability in a long-term sense, rather than giving provable delivery guarantees for each message. On the other hand, it still relies both on proofs of correct behavior to establish reputations, and trusted witnesses to determine and keep track of them. The reputation system in [3] does away with trusted witnesses and proofs in favor of self-rating groups of remailers. Remailers are arranged in cascades (fixed-order routes of determinate length). New cascades are formed at a regular interval, e.g., one day, and the formation of cascades is based on a communally generated random value so that no set of collaborating remailers can predict which remailer will be in which cascade, as long as at least one of them is honest. Remailers send test messages through their own cascades and can also receive evidence of failure from client senders. Rather than depending on proofs of remailer performance, a cascade fails when and only when some member of that cascade has declared it to have failed. All members of cascades that do not fail during an interval increase in reputation; all members of cascades that fail decrease. To make it harder for the head remailer of a cascade to undetectably fail or fail selectively, each of the cascade members is responsible for a portion of the messages that go through a cascade in each batch. In effect, each member is the head for some of the messages. Similarly, the tail of the cascade sends each message in a batch to each of the other cascade members rather than just directly to the recipient. All then attempt to deliver the message to the recipient. (Efficiency of communication for final delivery can be improved by using receipts when that is feasible.)

In both systems just described, it was necessary to redesign the remailer protocol so we could track remailer behavior. In the first case, we added receipts for each delivery to each remailer and to the ultimate destination, and we added trusted witnesses to verify delivery and record success or failure. In the second case, we used the usual remailer cascade protocol for messages inside the cascade, but we

introduced a new protocol for sending messages into the cascade and delivering messages from the cascade.

An Example: Anonymous Publishing

Free Haven [2] describes a design for a publishing system that can resist the attempts of powerful adversaries to find or destroy any stored data. It provides anonymity for readers and publishers, and pseudonymity (privacy and deniability) for the servers that store and serve the documents. Unlike related designs such as Freenet, the publisher of a document — not the servers holding the document — determines its lifetime. To counter malicious or flaky servers, publishers break documents up into shares, any adequately sized subset of which is sufficient to reconstruct the document. Servers then trade these shares around, allowing for servers to join and leave smoothly and also providing a moving target for an adversary hunting a particular share.

To prevent selfish or malicious users from filling up the available disk space, all who would publish must also provide servers, and servers form contracts to store each other's material for a certain period of time. Because servers can cheat and drop data early, Free Haven employs a reputation system to limit the damage done by servers that misbehave. Successfully fulfilling a contract increases a server's reputation and thus its ability to store some of its own data on other servers. This gives an incentive for each server to behave well — that is, as long as cheating servers can be identified.

In such a dynamic and anonymous environment, it is very difficult to reliably notice if a server drops data early. We might give the original publisher the task of monitoring the availability of his documents; he can then broadcast a claim that a particular document has been dropped early. If we don't want to rely on the original publisher, we can assign a random server as a "shepherd" for a document. Or for a more dynamic solution, we can use a "buddy system", where publishers put in two copies of each share, and the copies watch each other and broadcast a complaint if either disappears.

Anyone wishing to claim misbehavior can present a signed contract as an indication the file should have been kept, but there are a number of special cases where a signed contract is not sufficient proof to pinpoint a particular server as the culprit, such as if the server traded the share away before the contract expired. Because a claim cannot be taken as absolute proof, servers are left with the grim task of trying to determine the credibility of the claimer: if two or more servers disagree, onlookers need a way to determine which to believe. Keeping track of all claims ever made and tracking their results might give a server enough information to make reasonable guesses about the validity of each claim. This approach gets complex very quickly, and leaves lots of holes for a smart adversary.

Providing a way to verify claims in Free Haven remains an open problem. Given our experience designing the remailer reputation systems above, it seems most promising to redesign the entire system from the ground up, this time with verifiable claims in mind. Designing the original system without a clear idea of the reputation requirements made Free Haven's design significantly more complex and brittle.

It's tempting to scale down the reputation system instead. That is, servers use only local information from their own transactions and trades. In this scenario, a new server must separately convince each server it encounters that it is reputable before that server will allow it to publish any data. Because news about good and bad behavior does not propagate, servers must be more conservative about offering resources to unfamiliar nodes. To prevent stabilization into a static network with a small number of nodes trading data among each other and ignoring newcomers, many servers must explicitly risk re-

sources on new nodes. Rather than the global gossip system where the whole world learns from a server's first interaction, now each new server has a chance to waste the time and resources of every other server (the "screw every server once" attack).

Perhaps the most interesting point of Free Haven reputations is that servers use their reputation capital to obtain proportional resources from other servers. In order to build up to a reputation that allows a server to store a certain amount in the system (think of it like a credit limit), the server must previously have successfully stored that same size of other documents. Each server is able to commit at most his current credit limit in new contracts, and successfully completing a contract raises his credit limit. Because a server can cheat on at most the amount of useful work he has already done, each server is forced to perform at least 50% useful work. We will say more about this notion of "spending" reputation and quantifying risk below.

Reputation and Anonymity

Although the examples described above associate non-transferable reputation with long-lived nyms, there are many ways to vary this formula. (Think of a nym as a name.) Entities may be anonymous or have only short-lived names; reputations may be short-lived or transferable; and bindings between nyms, entities, and reputations may be varied.

We have already noted that in Free Haven, a server can both acquire and spend reputation. A central register or registers can keep track of each server's current reputation credit level, or each server can maintain its own view of the system. It is a small step from here to consider systems in which reputation can be paid in the form of coins or tokens. With such a system, the same entity can maintain reputation even while changing nyms. An entity that holds different nyms in different contexts can benefit from this feature — either to preserve reputation independent of private authentication key compromise, or to maintain perfect forward anonymity (future compromises linking a nym to an entity will not reveal previous transactions by the entity bearing that nym, even if all past behavior under all nyms is logged [4]).

Along with the advantages of fungible reputation come some potential problems. For example, an entity can obtain reputation in one context where it has functioned well and spend it in another context where it has not. Or the entity can transfer reputation to another entity entirely. This capability might be controlled by using cryptographic techniques that, e.g., bind all the reputation to the same nym or the same entity without revealing which nym (or entity) that is. But there are other concerns for this approach. For example, how do you diminish reputations for bad performance in such a model? (If it is possible to bind reputation tokens to an entity in an anonymity preserving way, it may also be possible to bind them to a duration so that the amount of reputation can be evaluated with respect to the time that the reputation bearer has existed. Thus we can distinguish longstanding, low-performing entities from untested ones.)

On the hand, in some contexts transferable reputation may not be a problem. For example, in a system like Free Haven that "pays" you for good performance with system services from others, as long as the amount of service credit taken from the system is no more than the amount of service provided to it, does it matter if that credit is transferred to others?

Conclusion: New Directions, Misdirections, and Other Questions

Reputation systems are already gathering momentum at the grassroots of the Internet. Special-purpose systems — some more ad hoc than others — have already been rolled into online auction services, messaging protocols, and online discussion sites. But despite this growing body of experience at building simple reputation systems and designing complex ones, the questions remain as intriguing as the solutions.

How can we fine-tune a reputation system in response to a specific threat model? In a relatively low-threat environment (e.g., tracking ISP uptime), ordinary statistical models will suffice. But most statistical models assume that data is biased at worst, not maliciously chosen by an adversary who wants us to make a particular decision. At this point, the emphasis in current research shifts from predicting behavior to minimizing risk. Is it really necessary to abandon statistical rigor? The field of machine learning has a rich history and a lot of experience at solving related problems. Is there some way to adapt these solutions to an adversarial context?

Similarly, what can we do when statements aren't verifiable, and where an adversary can either lie about real interactions, or fabricate spurious interactions and lie about those? We could try to make credibility charts and weight statements by credibility — but a smart adversary could try to trick our credibility calculations as well. If somebody finds a way to establish bounds on malicious influence on such a system, the range of problems we can solve with reputation would explode overnight.

We have already seen how reputation can enhance privacy. Can we go one step farther, and assign reputation based on an expectation of protecting privacy? In distributed privacy systems, the privacy provided is typically based on an assumption that some subset of system components will both perform duties correctly *and* will not reveal some parts of their data and/or operations. A privacy destroying adversary might offer very reliable service while using information from various compromised system elements to compromise privacy. Can we say anything meaningful about reputations based on reliability at keeping secrets, or are we limited to making statements about the probability of privacy compromise given the likelihood and structure of component compromise? Would knowledge of such reputation help us design more reliable privacy systems?

What if we treat reputation as currency? Currency implies economics. How do you get reputation currency? Do currency-based approaches always imply transitive trust? Is the supply of reputation currency constant, increasing, or decreasing? Does it expire, or slowly lose value over time? Where, ultimately, does a currency come from — a decentralized Federal Reserve? Does it materialize as a side-effect of performing work from the system? Is there credit? Or does currency only appear when the system is bootstrapped, and if so, how? Is currency global, or do individual servers mint their own currencies? Can we evolve a viable market that is scalable based on these local currencies issued by individual servers? Can a global currency be bootstrapped from local currencies?

The number and diversity of real-world reputation systems is staggering. Through a byzantine mass of credit reports, product reviews, earnings statements, flat-out gossip, and a thousand other information channels, we try to convince one another of our honesty and competence. In the process we expose far more detail about ourselves than we might wish. Online reputation systems promise to be still more complex and difficult to build than their real-world analogues, but they hold out the promise of enabling decentralized interaction and protecting privacy in ways that today's systems of trust cannot.

Footnotes

[1] R. Dingledine, M.J. Freedman, D. Hopwood, and D. Molnar. “A Reputation System to Increase MIX-Net Reliability”, Information Hiding, 4th International Workshop, I.S. Moskowitz (ed.), Springer-Verlag, LNCS 2137, April 2001.

[2] R. Dingledine, M.J. Freedman, and D. Molnar. “The Free Haven Project: Distributed Anonymous Storage Service”, Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, H. Federrath (ed.), Springer-Verlag, LNCS 2009, July 2000.

[3] R. Dingledine and P. Syverson. “Reliable MIX Cascade Networks through Reputation”, Financial Cryptography, 6th International Conference, M. Blaze (ed.), Springer-Verlag, LNCS, March 2002.

[4] P. Syverson, S. Stubblebine, and D. Goldschlag. “Unlinkable Serial Transactions”, Financial Cryptography, 1st International Conference, R. Hirschfeld (ed.), Springer-Verlag, LNCS 1318, February 1997.