A Sybil-proof one-hop DHT

Chris Lesniewski-Laas
MIT CSAIL
Cambridge, MA, USA
ctl@mit.edu

ABSTRACT

Decentralized systems, such as structured overlays, are subject to the Sybil attack, in which an adversary creates many false identities to increase its influence. This paper describes a one-hop distributed hash table which uses the social links between users to strongly resist the Sybil attack. The social network is assumed to be *fast mixing*, meaning that a random walk in the honest part of the network quickly approaches the uniform distribution. As in the related SybilLimit system [25], with a social network of n honest nodes and m honest edges, the protocol can tolerate up to $o(n/\log n)$ *attack edges* (social links from honest nodes to compromised nodes). The routing tables contain $O(\sqrt{m \log m})$ entries per node and are constructed efficiently by a distributed protocol. This is the first sublinear solution to this problem. Preliminary simulation results are presented to demonstrate the approach's effectiveness.

1. INTRODUCTION

Decentralized systems on the Internet are vulnerable to the "Sybil attack", in which an adversary creates numerous false identities to influence the system's behavior. [8] This problem is particularly pernicious when the system is responsible for routing messages amongst nodes, as in the Distributed Hash Tables (DHT) [24] which underlie many peer-to-peer systems, because an attacker can prevent honest nodes from communicating altogether. [23] If a central authority certifies identities as genuine, then standard replication techniques can be used to fortify these protocols. [4, 19] However, the cost of universal strong identities may be prohibitive. Instead, recent work [26, 25, 6, 18, 14, 5] proposes using the weak identity information inherent in a social network to produce a completely decentralized system. If the adversary can only convince a small fraction of honest users to form trusted social links with his Sybil identities, then the honest users can protect themselves against a Sybil attack.

This paper poses the problem: *given a well-connected social network, can honest nodes reliably and efficiently communicate in spite of the adversary's deception?* Section 2 reviews some of the related work on this subject. Section 3 presents the first sublinear protocol which solves the problem, and then extends this simple protocol into a highly efficient but more complicated solution. Section 4 gives preliminary simulation results showing the effectiveness of the approach. The remainder of this introduction expands on the problem and its implications for decentralized systems.

1.1 The Byzantine Dissidents Problem

Imagine that there is network of dissident free-thinkers (called *honest nodes*) in the Byzantine Empire, a nation controlled by a ruthless junta. [12] The dissidents have no official leader, since the regime can identify and remove any such person, but they are connected by a network of individual trust relations formed through collaborations and introductions. These social links are assumed to be reflexive (undirected), and each dissident keeps track of his immediate friends, so they are always in contact. The social network is assumed to be well-connected (this requirement will be made more rigorous in Section 3).

The regime employs a number of spies (*Sybil nodes*) who infiltrate the network by gaining the trust of honest nodes. A link between an honest node and a Sybil node is called an *attack edge*. [26, 25] Honest nodes cannot distinguish between attack edges and honest edges, and furthermore, spies can create an arbitrary number of connections to an arbitrary number of other spies (the regime's Sybil identities). The "Sybil region" controlled by the regime is indistinguishable from the region of honest nodes except that the number of links between the two regions (the number of attack edges, g) is smaller than the number of links between any two similarly-sized subsets of the honest region.

The regime does not overtly interfere with communication between immediate friends, but its spies aim to disrupt the network by spreading misinformation. For example, imagine that the honest node Sergei wants to contact the honest

SocialNets'08, April 1, 2008, Glasgow, Scotland, UK.

node Tanya (e.g., to get a copy of a learned treatise Tanya has written), and Sergei does not have a direct social link with Tanya. (Assume that Sergei can recognize Tanya as genuine when he reaches her, through, e.g., self-certifying identities. [15, 16])

Sergei might ask his friend Umberto for help, but if Umberto is a spy, he might mislead Sergei. For example, Umberto could send Sergei on a wild goose chase via his Sybil identities Uma, Upton, Ulysses, Ursula, and so on, consuming Sergei's time without bringing him any closer to Tanya. [23]

As pointed out in [6], Sergei has a simple and correct protocol to find Tanya: he can ask each of his friends Umberto, Valerie, Walter, etc., each of whom must ask each of their friends, and so on until the request floods the entire social network. This guarantees that any connected pair of honest nodes will eventually find each other, but costs $\Omega(n)$ messages per query, which may become unsustainable as the size of the network increases. The aim of Section 3 will be to present a novel protocol which dramatically reduces this overhead.

1.2 The Byzantine Dissidents' relevance

The problem of the Byzantine Dissidents arose from the study of Sybil attacks against DHTs: a solution is a Sybil-resistant DHT routing protocol. [6, 23] Since DHTs are a critical layer of many decentralized or peer-to-peer network systems, a technique for constructing Sybil-resistant DHTs is of great importance.

That the above story is about dissidents is not accidental; a solution is essentially a censorship-resistant system. [21] As such, this paper's protocols may be useful as a part of publication systems for real-life dissidents and whistleblowers. Since this paper makes no provisions for anonymity, such features would need to be added using techniques such as onion routing. [7, 9]

Finally, the "social network" used by the protocol need not be limited to the connections between human friends. It could be, instead, a physical network such as the Internet or a wireless ad-hoc network. In this context, the Byzantine Dissidents problem may be applicable to the problems of secure, fault-tolerant, and accountable routing.

2. BACKGROUND

Shortly after the introduction of scalable peer-to-peer systems based on DHTs, the Sybil attack was recognized as a serious security challenge. [8, 13, 23, 22] A number of techniques [4,19,22] have been proposed to make DHTs resistant to a small fraction of Sybil nodes, but all such systems ultimately rely on a certifying authority to perform admission control and limit the number of Sybil identities. [8, 20, 3]

Several researchers [14,18,6,5] proposed using social network information to fortify peer-to-peer systems against the Sybil attack. The *bootstrap graph* model [6] introduced a correctness criterion for secure routing using a social network and presented preliminary progress towards that goal, but left a robust and efficient protocol as an open problem.

Recently, the SybilGuard and SybilLimit systems [26, 25] have shown how to use a *fast mixing* social network as a defense against the Sybil attack in general decentralized systems. Using SybilLimit, an honest node can certify other players as "probably honest", accepting no more than $O(\log n)$ dishonest Sybil identities per attack edge. (Each certification costs $O(\sqrt{m})$ bandwidth, where m is the number of edges between honest nodes.) For example, SybilLimit's vetting procedure can be used to check that at least one of a set of storage replicas is likely to be honest.

Applying SybilLimit naively to the Byzantine Dissidents problem yields a protocol which costs $O(n^2\sqrt{m})$ bandwidth to protect against a maximum of $o(n/\log n)$ attack edges.[1] Unfortunately, this is more costly than the simple flooding protocol described above. However, the next section shows how to adapt the underlying techniques developed for SybilLimit to produce an efficient protocol for the Byzantine Dissidents problem.

3. PROTOCOLS

3.1 System model

The system model is the same as in SybilLimit [25], summarized briefly here. The social network has m undirected edges connecting all n honest nodes. The entire graph is not stored at any one machine, but each node knows its immediate neighbors in the social network. Honest nodes are assumed to have unforgeable locally-generated public/private key pairs, and all nodes know their immediate friends' public keys (through, e.g., a physical rendezvous). When an honest node searches for another honest node, it can distinguish the target from an imposter through this public key, a self-certifying identifier, or some other out-of-band means.

The *mixing time*, w, is a measure of the connectivity of the social network: highly connected networks have low mixing time. It is the number of steps a random walk from an arbitrary node must take before approximating the stationary distribution of the network. In other words, a random walk of at least w steps will end at an (approximately uniformly) random edge in the network. Honest nodes are assumed to know a rough upper bound on w. The social network is said to be *fast mixing* if $w = O(\log n)$; real-life social networks are likely to be fast mixing. [25]

All Sybil nodes are controlled by a Byzantine adversary, who can create arbitrary edges between Sybil nodes. (An "honest" node whose software's integrity has been compromised by the adversary is considered a Sybil node.) Sybil nodes may deviate from the protocol by failing to respond to requests, delaying responses, or responding with bad data.

[1] As a reminder, the asymptotic notation $O(f(n))$ means "grows no faster than $f(n)$", $o(f(n))$ means "grows more slowly than $f(n)$", $\Omega(f(n))$ means "grows at least as fast as $f(n)$", and $\Theta(f(n))$ means "grows at the same rate as $f(n)$".

	Bound	Description	Explicitly known?
n	none	# honest nodes	implicit
m	$\Omega(n), O(n^2)$	# honest edges	implicit
w	$O(\log n)$	mixing time	rough upper bound
g	$o(n/\log n)$	# attack edges	implicit
r	$\Theta(\sqrt{m \log m})$	finger table size per node	estimated by protocol

Figure 1: Numeric parameter reference

The adversary can observe messages between honest nodes, but cannot modify them or block communication entirely.

There are g *attack edges* between Sybil nodes and honest nodes. The honest nodes do not know g. If the social network is fast mixing, then a random walk of length w will cross an attack edge with *escape probability* $O(\frac{g \log n}{n})$. [25] This paper assumes g is bounded by $o(n/\log n)$. This limits the escape probability to $o(1)$, and ensures that the random walk stays within the *honest region* with substantial probability $1 - o(1)$.[2] If g is greater than the bound $o(n/\log n)$, then the protocols cannot provide strong guarantees, although the preliminary simulations in Section 4 suggest that the method is robust to a much larger fraction of attack edges.

3.2 A simple sublinear protocol

The protocol consists of two phases. In the first phase, nodes exchange messages to construct local routing tables of size $r = \Theta(\sqrt{m \log m})$. (The reason for this value will be explained below.) In the second phase, nodes use the constructed tables to search for other nodes.

Each honest node starts with its social links and a rough upper bound on w. To construct routing tables, each node initiates r independent random walks of length w, recording the final node in each walk as a "finger" in its routing table.[3] Since each walk's escape probability is bounded by $o(1)$, at least $\Omega(r)$ of the finger table entries will be honest.

By the fast mixing property, the last edges traversed by a node's random walks are a nearly-uniform sample of the honest region's edges. Thus, each node appears in other nodes' finger tables with frequency proportional to its degree.

Routing is simple: to perform a search, a node s broadcasts the target t's identifier to all of s's fingers $u_1, \ldots, u_r$. (See Figure 3.) At least $\Omega(r)$ of the fingers u_i are honest, and at least one of the honest u_i has the target t as a finger. This u_i can forward the request onward to t, or return t's address to s.

More specifically, for any honest u_i, the target is in u_i's finger table with probability at least $\Omega(r)/m$. Therefore, the probability that *none* of the honest u_i have the target as a finger is bounded by

$$p_{fail} = \left(1 - \frac{\Omega(r)}{m}\right)^{\Omega(r)} = O\left(\left(1 - \frac{\log m}{r}\right)^{r}\right)$$
$$= O(e^{-\log m}) = O\left(\frac{1}{m}\right)$$

This explains the $\sqrt{\log m}$ factor in r: with high probability, some finger will have the target node in its own finger table. If r were instead $\Theta(\sqrt{m})$, the protocol would only provide a constant probability of finding the target node.

Identifier	IP address
SHA1(PK_1)	18.26.4.9
SHA1(PK_2)	208.77.188.166
⋮	⋮
SHA1(PK_r)	130.209.34.12

Figure 2: Example finger table

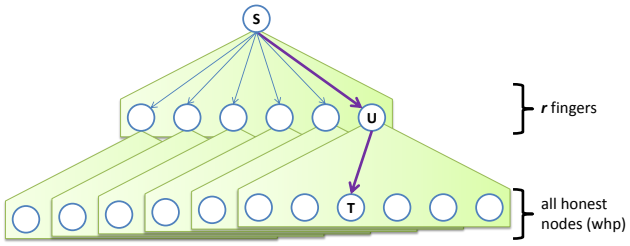

Figure 3: Routing from s to t via s's finger u

3.3 On estimating w and r

The parameters n, m, w, g, and r are defined from a "God's-eye view" of the social network; since real nodes cannot distinguish Sybil nodes from honest nodes, they cannot know their precise values. However, to run the above protocol, each node needs only a rough upper bound on the mixing time w and a rough lower bound on the required table size r.

Too small an estimate of w will prevent random walks from reaching the stationary distribution; a larger value will cause more walks to escape from the honest region. Since $w = O(\log n)$, it suffices to choose a w in the right ballpark for the number of honest nodes in the social network: if there are roughly 10^6 nodes, then w should be approximately 20. If the number of honest nodes is entirely uncertain, there's nothing wrong with running multiple instances of the protocol for every value of w up to, e.g., 30, supporting networks of up to billions of nodes. This need not require extra bandwidth, since the prefixes of a random walk of length w are themselves random walks of length $1, 2, \ldots, w-1$.

Too small an estimate of r will cause lookups to fail. In the SybilLimit [25] system, overestimating r would lead to

[2] For a small fraction ϵ of honest nodes which are near a concentration of attack edges, this guarantee does not hold. See [25] for details.

[3] Note that this step can be computed by communicating only between neighbors (see [25] for details). Thus, for a node of degree d, the number of messages is only $O(dw) = O(d \log n)$, while the number of bits transmitted is still $\Theta(r) = \Theta(\sqrt{m \log m})$.

admitting too many Sybil nodes.[4] However, this paper's protocols need only a lower bound on r: overestimating r cannot harm correctness, since it only increases the likelihood that two honest nodes share a finger. Therefore, nodes simply continuously increase their table size r until the majority of routing requests succeed.

To check whether the table is sufficiently large yet, a node may sample $\Omega(\log n)$ random nodes by taking random walks of length w. If fewer than some fixed fraction of the sampled nodes are already reachable through the routing protocol, the table is not yet large enough, and the estimate of r must be increased (e.g., by doubling it).[5] Once r is at least $\Omega(\sqrt{m \log m})$, all searches for random honest nodes will succeed with high probability (although an $o(1)$ fraction of the sampled walks will cross an attack edge and thus can be forced by the adversary to fail).

3.4 Reducing lookup to $O(1)$ messages

The simple routing protocol above requires $O(\sqrt{m \log m})$ messages per lookup; while this is the first sublinear solution to the Byzantine Dissidents problem, it is still not ideal. A somewhat more complex protocol based on one-hop DHTs [10, 11] can reduce this overhead dramatically.

As in Chord and other DHTs, each node u generates a random ID $\mathcal{H}(u)$ on the unit circle $[0, 1)$ by applying a hash function to its public key. (Assume that the adversary cannot influence his Sybil nodes' random IDs; this assumption will be relaxed later.) As before, in the first phase, each node initiates r random walks and collects the final nodes into a finger table (sorted by ID).

In the second phase, each node u constructs a *successor table* consisting of the $\Theta\left(\frac{n \log n}{r}\right) = O(r)$ nodes immediately following $\mathcal{H}(u)$. This table is similar to Chord's successor table, but is differently constructed. Chord maintains its successor tables by a stabilization protocol amongst the successor sets on the ring, which is subject to manipulation by malicious successors. Instead, the node u sends a query to each of its *fingers* v, requesting v's $k = \Theta(\log m)$ fingers immediately following $\mathcal{H}(u)$.[6] Sending these requests to fingers is better than to successors because a substantial fraction of u's fingers are guaranteed to be honest; there is no such guarantee about u's successors.

The argument for correctness of the successor table construction is similar to the argument for correctness of the simpler protocol of Section 3.2. At least $\Omega(r)$ of u's fingers are honest, and these honest fingers collectively know all of u's honest successors. (Indeed, they collectively know the entire set of honest nodes.) Moreover, consider one of u's successors w, and an honest finger v which knows w. The distance between $\mathcal{H}(w)$ and $\mathcal{H}(u)$ on the ring is less than $O\left(\frac{\log n}{r}\right)$. Since v's r fingers are uniformly distributed on the ring, it's very unlikely that more than $k = \Theta(\log m)$ fall in this $O\left(\frac{\log n}{r}\right)$ ID range, so v *will* tell u about w. Therefore, with high probability, the node u will learn about all of its nearest $\Theta\left(\frac{n \log n}{r}\right)$ honest successors.

Using the finger and successor tables, routing now proceeds typically for a one-hop DHT. Given the target node's identifier $\mathcal{H}(t)$, the requester s chooses the finger with $\mathcal{H}(v)$ immediately preceding $\mathcal{H}(t)$ and sends the request $\mathcal{H}(t)$ to the finger v. With high probability, the number of identifiers between $\mathcal{H}(t)$ and $\mathcal{H}(v)$ is less than $\frac{n \log n}{r}$. Thus, if the finger v is honest, it will have the target node in its successor table and can forward the request onwards (or return t's address to s).

If the finger is dishonest, it may return a failure result to s, or simply refuse to forward the request. However, the requester can simply continue on to try the next nearest predecessor to $\mathcal{H}(t)$ in its finger table. Since only an $o(1)$ fraction of the fingers are dishonest, the expected number of tries to find the target is constant, and thus the message complexity of the lookup protocol is $O(1)$.

The maximum number of tries needed to reach the target will be $O(\log n)$, with high probability. In a situation where it's impossible for s to detect failure, or if $O(\log n)$ maximum latency is too long, s can simply send the request simultaneously via all $O(\log n)$ predecessor fingers. At least one of them will be honest and able to forward the message to the target.

3.5 Finger table balancing protocol

The structured protocol above assumes that the adversary cannot influence its nodes' random IDs. However, in reality, a fixed hash function over private keys does not provide this property: an adversary can repeatedly generate new private keys until he gets one that hashes into a particular range.

The adversary may attempt to isolate an individual node by generating many Sybil IDs near the victim's ID, causing its true predecessors to fill up their successor tables, and forcing any node searching for the target to try many bad fingers before finding a good one. [23, 22] This situation can be detected by honest nodes as suspicious, since highly clustered IDs are very unlikely to happen by chance. However, it is challenging to distinguish the dishonest fingers in a cluster, which should be discarded, from the honest fingers being attacked.

The key is to balance each node's finger tables so that the fingers' IDs are evenly spaced around the unit circle. Chord balances load by assigning each node $k = \Theta(\log n)$ virtual IDs generated by independent hash functions $\mathcal{H}_1, \ldots, \mathcal{H}_k$, and then running the routing protocol over the virtual IDs. [24]

[4] SybilLimit's solution is a subtle sampling-based protocol which each honest node uses to estimate r.

[5] The sampled nodes may themselves be added to the routing table as fingers, reducing the overhead of the sampling protocol.

[6] Naively, this addition seems to increase the routing table's size by a logarithmic factor k. However, note that an average edge is known by $O(\log m)$ honest fingers: therefore, the sets returned by the honest fingers should be redundant by a logarithmic factor. Dishonest fingers may contrive to return many unique successor IDs, but this imbalance is easily detected and the extra successors discarded.

However, simply adding virtual IDs doesn't solve the problem: an adversary can eclipse a targeted node by generating Sybil IDs near *all* of the targeted node's virtual IDs.

The proposed solution is inspired by cuckoo hashing [17], and is similar to a scheme in [2]. Briefly, each node is given $k = \Theta(\log n)$ virtual IDs, and abnormally clustered virtual IDs are discarded from the finger table until it is evenly spaced. With high probability, each honest finger will retain at least one virtual ID.

More specifically, let $\mathcal{H}_i$ be a family of k hash functions, such that it is computationally infeasible for the adversary to find inputs x and y such that $\mathcal{H}_i(x)$ is near $\mathcal{H}_i(y)$ for *all* $\mathcal{H}_i$. (In this context, "near" means the distance on the unit circle is less than $1/r$.) A node's k virtual identities (vIDs) are defined to be $\mathcal{H}_i(PK_u)$, where PK_u is the node's public key (or hash thereof). By $\mathcal{H}_i$'s definition, it is difficult for the adversary to compute a public key whose vIDs collide with all of another node's vIDs.

After a node has done r random walks and collected the virtual identities of its r fingers, it balances its finger table by pruning clustered vIDs. For each pair of fingers with vIDs closer than $\Theta(1/rk)$, the node picks the finger with more remaining vIDs and discards that vID. Very few pairs of honest fingers will be pruned: if one imagines dividing the unit circle into bins of size $\Theta(1/rk)$, this procedure is essentially the same as cuckoo hashing. [17] Thus, the most full bin will only contain $O(\log \log nk)$ honest vIDs.

The adversary may contrive to cause an honest finger to be discarded by choosing many vIDs near the target finger's vIDs. However, the pruning priority rule prevents this attack. Consider an honest finger with only one vID remaining: that finger will always win against an adversary's vID, unless the adversary's vID collides with the honest finger on all $\mathcal{H}_i$.

Since vIDs can be computed from public keys on-the-fly using the hash family $\mathcal{H}_i$, in theory this modification does not increase the finger table size. However, fast access to the table will require it to be sorted by vID, and so the $k = \Theta(\log n)$ virtual IDs per node will effectively increase the protocol's storage requirements by a factor of k. The rest of the protocol remains unchanged from Section 3.4.

4. PRELIMINARY RESULTS

The simple protocol of Section 3.2 has been implemented as a simulation proof-of-concept and tested against a graph extracted from the Orkut social networking service. [1] Following SybilLimit [25], the graph was preprocessed to remove nodes with degree less than 5 and disconnected clusters. The remaining graph has 7335 nodes and 56211 edges; node degrees follow a power law distribution. The simulation used the parameters $w = 10$ and $r = 1000$; since random walks sometimes collide, the typical finger table sizes were around 650 entries.

To generate an instance with g attack edges, the simulator marked random nodes as "evil" until there were g edges between marked nodes and non-marked nodes. For example, in the instance with $g = 32000$, there are $n = 4487$ honest nodes (with $m = 29445$ honest edges) and 2848 Sybil nodes. The adversary's behavior is to swallow all escaping random walks and ignore all requests: this is the optimal attack against the simple broadcast-based protocol.

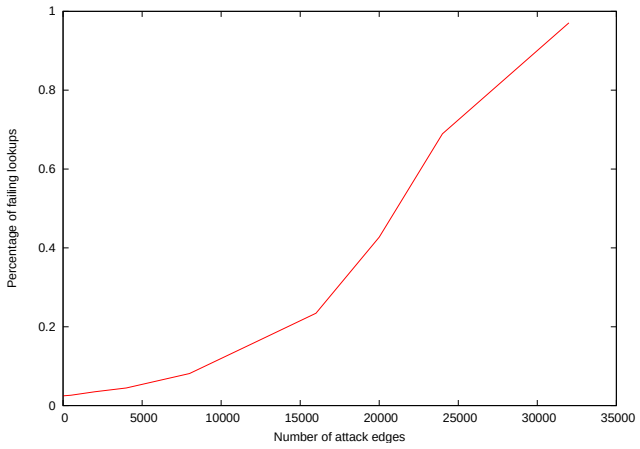

Figure 4: Routing failure rate versus attack edges

Figure 4 plots the rate of lookup failures versus the number of attack edges g. Note that the protocol provides a strict guarantee only up to $g = o(n/\log n)$, or approximately 750 attack edges. The graph shows the protocol's graceful degradation when the adversary is very powerful. In practice, the number of failures is largely unaffected by the adversary until g becomes substantial compared to the number of honest edges (e.g., with $g = 12000$ and $m = 43930$, the failure rate is 15.8%).

5. CONCLUSION

This paper presents the first structured DHT routing layer which uses a social network to provide strong resilience against a Byzantine Sybil attacker with many attack edges. Routing table sizes are typical of one-hop DHTs, at $O(\sqrt{m \log m})$ entries per node. Lookups take only $O(1)$ messages, and lookup load is distributed evenly amongst nodes proportionally to their degree.

This protocol solves the *Byzantine Dissidents problem*: it permits a decentralized network of honest confederates to maintain coordinated communication despite active interference and misinformation spread by an attacker. This problem has applications to decentralized Internet system design, censorship resistance, and secure network routing.

The technique presented here only applies straightforwardly to a one-hop structured DHT. Multiple-hop routes pose a challenge, because each hop has a non-negligible chance of being controlled by the adversary; after $O(\log n)$ hops, almost all routes will be tainted. This paper leaves open the question of whether a structured DHT with logarithmic table size can be made highly Sybil-resistant using social networks.

Acknowledgements

I would like to thank Frans Kaashoek and the anonymous reviewers for their many helpful comments. This research is sponsored by the National Science Foundation under Cooperative Agreement ANI-0225660 (Project IRIS).

6. REFERENCES

[1] Orkut online community. http://orkut.com/.

[2] B. Awerbuch and C. Scheideler. Towards scalable and robust overlay networks. *IPTPS Workshop,* Bellevue, WA, Feb. 2007.

[3] N. Borisov. Computational puzzles as sybil defenses. *Peer-to-Peer Computing,* pages 171-176, 2006.

[4] M. Castro, P. Druschel, A. J. Ganesh, A. I. T. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. *OSDI,* Boston, MA, Dec. 2002.

[5] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. *Applications, Technologies, Architectures, and Protocols for Computer Communication,* pages 128–132, ACM Press New York, NY, USA, 2005.

[6] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. J. Anderson. Sybil-resistant DHT routing. *ESORICS,* pages 305-318, 2005.

[7] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. *USENIX Security,* pages 303-320, San Diego, CA, Aug. 2004.

[8] J. R. Douceur. The sybil attack. *IPTPS Workshop,* pages 251-260, Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, ed. Springer Lecture Notes in Computer Science 2429, Cambridge, MA, Mar. 2002.

[9] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. *CCS,* pages 193-206, Vijayalakshmi Atluri, ed. ACM, Washington, D.C. Nov. 2002.

[10] A. Gupta, B. Liskov, and R. Rodrigues. Efficient routing for peer-to-peer overlays. *NSDI,* pages 113-126, San Francisco, CA, Mar. 2004.

[11] I. Gupta, K. P. Birman, P. Linga, A. J. Demers, and R. van Renesse. Kelips: Building an efficient and stable p2p DHT through increased memory and background overhead. *IPTPS Workshop,* pages 160-169, M. Frans Kaashoek and Ion Stoica, ed. Springer Lecture Notes in Computer Science 2735, Berkeley, CA, Feb. 2003.

[12] L. Lamport, R. E. Shostak, and M. C. Pease. The byzantine generals problem. *ACM ToPLaS,* 4(3):382-401, 1982.

[13] B.N. Levine, C. Shields, and N.B. Margolin. A survey of solutions to the sybil attack. University of Massachusetts Amherst, Amherst, MA, 2006.

[14] S. Marti, P. Ganesan, and H. Garcia-Molina. DHT routing using social links. *IPTPS Workshop,* pages 100-111, Geoffrey M. Voelker and Scott Shenker, ed. Springer Lecture Notes in Computer Science 3279, La Jolla, CA, Feb. 2004.

[15] D. Mazières, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating key management from file system security. *SOSP,* pages 124-139, Kiawah Island, SC, Dec. 1999.

[16] P. Nikander and J. Laganier. An IPv6 prefix for overlay routable cryptographic hash identifiers (ORCHID). RFC 4843, 2007.

[17] R. Pagh and F. F. Rodler. Cuckoo hashing. *J. Algorithms,* 51(2):122-144, 2004.

[18] B. C. Popescu, B. Crispo, and A. S. Tanenbaum. Safe and private data sharing with turtle: Friends team-up and beat the system. *Security Protocols Workshop,* pages 213-220, 2004.

[19] R. Rodrigues and B. Liskov. Rosebud: A scalable byzantine-fault-tolerant storage architecture. MIT CSAIL, Technical Report TR/932, Dec. 2003.

[20] H. Rowaihy, W. Enck, P. McDaniel, and T. L. Porta. Limiting sybil attacks in structured p2p networks. *INFOCOM,* pages 2596-2600, 2007.

[21] M. W. A. D. Rubin and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. *USENIX Security,* pages 59–72, August 2000.

[22] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. *INFOCOM,* 2006.

[23] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. *IPTPS Workshop,* pages 261-269, Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, ed. Springer Lecture Notes in Computer Science 2429, Cambridge, MA, Mar. 2002.

[24] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *ToN,* 11(1):17-32, 2003.

[25] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. A near-optimal social network defense against sybil attacks. To appear. *IEEE Symposium on Security and Privacy,* Oakland, CA, May 2008.

[26] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending against sybil attacks via social networks. *SIGCOMM,* pages 267-278, Piza, Italy, Sept. 2006.