Secure Peer-to-peer Networks for Trusted Collaboration

(Invited Paper)

Kevin W. Hamlen and Bhavani Thuraisingham
Computer Science Department – MS EC31
University of Texas at Dallas
800 W. Campbell Rd.
Richardson, Texas 75080-3021, USA
{hamlen, bhavani.thuraisingham}@utdallas.edu

***Abstract*—An overview of recent advances in secure peer-to-peer networking is presented, toward enforcing data integrity, confidentiality, availability, and access control policies in these decentralized, distributed systems. These technologies are combined with reputation-based trust management systems to enforce integrity-based discretionary access control policies. Particular attention is devoted to the problem of developing secure routing protocols that constitute a suitable foundation for implementing this security system. The research is examined as a basis for developing a secure data management system for trusted collaboration applications such as e-commerce, situation awareness, and intelligence analysis.**

I. Introduction

The advent of popular peer-to-peer (P2P) networks like Napster [1] and Gnutella [2] has heralded an explosion of interest in P2P network design both among researchers and practitioners. P2P networks have increased in popularity partly because they can be implemented atop a diverse collection of hardware and software, making them relatively inexpensive to deploy and maintain. The network infrastructure also tends to be highly fault-tolerant, and bandwidth and other computational resources tend to be well balanced across peers, making the network highly robust.

A robust network design requires that peers in a P2P network be considered semi-trusted or untrusted, so to ensure integrity and confidentiality of shared data it is critical that P2P networks be secure. In recent years there has been a vast array of research towards enforcing the security guarantees necessary to achieve system-wide, end-to-end security policies in P2P networks (c.f., [3], [4]). Recently we have designed the Penny system [5], which combines several of these advances to efficiently enforce strong data integrity policies in structured P2P networks.

This paper describes secure P2P networks and their support for building trusted applications. We first discuss our approach to enforcing integrity policies in Penny in Section II. Penny implements a reputation-based trust management system based on EigenTrust [6] in the context of a Chord network [7]. One of the most challenging aspects of developing a secure P2P network is establishing a secure routing structure over which messages and data can reliably be exchanged in the presence of malicious peers. In Section II-C we discuss some of the issues and open problems in this area.

P2P networks provide the infrastructure to support various technology applications such as data management, collaboration, and decision-making. These in turn support real-world applications including e-commerce, situation awareness, and intelligence analysis. Secure P2P networks can be used as a foundation for supporting trusted applications. In Section III we discuss some of our preliminary ideas on hosting a trusted data manager on the Penny system. In particular, we discuss the issues involved in decamping data management objects into multiple Penny objects so that the integrity policies can be enforced on the Penny objects. Supporting trusted collaboration is briefly discussed in Section IV and we conclude with a summary in Section V.

II. Secure Peer-to-peer Networks

A. Availability, Integrity and Confidentiality Vulnerabilities

P2P networks have developed as a means of evenly balancing the computational expense associated with delivering network services. In contrast to a traditional network, which divides its constituent hosts into servers and clients, P2P networks homogeneously treat all hosts as *servents*, assigning each both server and client functionality. This allows services to be delivered from a large number of servents rather than from a relatively small number of servers. For example, Napster, Inc. [1] achieved early commercial success using P2P technology to serve music content to users by storing most of that content on end-user machines rather than on centralized servers. This reduced costs, improved reliability, and greatly expanded the variety of content that they could offer. Subsequently, P2P has been used for general-purpose file-sharing in popular implementations such as Gnutella [2], KaZaA [8], LimeWire [9], and many others.

From a security standpoint, P2P networks ostensibly offer inherent robustness and availability properties not easy to achieve in a traditional network design. For example, an attacker wishing to effect a denial of service in a traditional network can focus her attack on a relatively small number of centralized servers, whereas in a P2P network the attacker must compromise a relatively large number of servents in order to fully disconnect the network.

However, in practice many P2P networks remain vulnerable to denial of service attacks because the homogeneity of the network results in greater interdependence among hosts. For example, in a Chord network [7], any pair of peers who wish to communicate must trust the $O(\log n)$ other peers who constitute the initial routing path between them through the network overlay (where n is the number of peers in the network). These hosts are chosen deterministically by the routing protocol, so to disconnect the two hosts it suffices to compromise any one of these $O(\log n)$ peers. In general, this means that compromising one host in a Chord network prevents numerous hosts from communicating even if it does not disconnect the entire network. Other protocols like CAN [10], Pastry [11], and Tapestry [12] have similar vulnerabilities of varying severity.

Many existing P2P networks also suffer from serious data integrity vulnerabilities because it is easy for peers in the network to lie to other peers about the data they serve. Peers can therefore spread corrupt content and malware merely by publishing it under a misleading name or with false keywords. Unsuspecting peers then download and propagate this low-integrity data to other peers. Such vulnerabilities are a major issue for real-world P2P implementations today. For example, two studies published in 2006 detected malware in as much as 68% of all executable content exchanged over KaZaA [13] and in 15% of all files exchanged over Limewire [14]. Integrity violations are therefore a significant concern for owners, administrators, and users of these networks.

Confidentiality is often cited by P2P users as an appealing advantage of P2P networks, but in reality strong confidentiality guarantees are deceptively difficult to attain. The confidentiality desired by P2P users typically comes in two forms: *Data confidentiality* policies prohibit the leaking of high-confidentiality, shared objects to low-privileged peers, while *user anonymity* policies prohibit the divulging of a user's private information. Such private information might include login credentials, a history of files shared or downloaded, or a list of the peers with which a user has interacted in the past.

Standard P2P network designs do not directly support either of these classes of security policies. Data confidentiality is not supported because shared objects are all public in today's P2P networks, and can therefore be downloaded freely by untrusted peers. User anonymity is not supported because without a central authority, login credentials and other private information must typically be divulged to a variety of other peers during authentication and while routing object lookup queries and other private messages through the network overlay.

Thus, although many popular P2P network implementations seem to offer both availability and confidentiality to users, enforcing strong availability, integrity, and confidentiality policies in P2P networks is a challenging domain of active research. The continuing growth of P2P networking as an ever more critical part of modern computing infrastructures, along with its appeal as a practical and cost-effective approach to load-balancing issues in large networks, argues that P2P security should be a high priority for cyber-security researchers.

B. Reputation-based Trust Management

Reputation-based trust management has emerged as an extremely promising technology for addressing many of these security vulnerabilities without sacrificing the load-balancing advantages of decentralization. A trust management system maintains a global *trust label* t_a for each agent a in the network. When the system is reputation-based, label t_a is an aggregation of the local opinions of all agents in the network based on their prior experiences with agent a. To compute t_a, each opinion of agent a is weighted by the reputation of the opiner, so that agents with good reputations are more influential than those with poor reputations or no reputation.

A goal of the trust management system is to allow only non-malicious peers to accrue good reputations with high probability. This allows non-malicious agents to easily identify malicious agents and potentially censor them from transactions. For example, a file served from a disreputable peer might be assigned a low integrity label by the receiving host. Similarly, the routing protocol might avoid forwarding messages via disreputable peers. Thus, tracking global reputations allows each peer to benefit from the experiences of all other peers in the network.

Although trust labels are global, they can be maintained in a decentralized setting via replication. For example, in the EigenTrust system [6] each agent's global trust label is tracked by k distinct peers (where k is a constant defined at network initialization). These k peers are referred to as the agent's *score-managers*. Peers report feedback to all k of agent a's score-managers after each transaction with agent a, thereby updating t_a. When agent a participates in many positive transactions, t_a therefore increases.

Label t_a can be retrieved by any peer by contacting all k score-managers and computing the median of their responses. Thus, subverting an agent's reputation requires subverting at least $k/2$ of the agent's score-managers, which is difficult when k is large. Score-managers of agent a are chosen by applying a secure hash function to agent a's IP number, so that agents can choose neither their score-managers nor the agents for whom they act as score-manager. This prevents a malicious collective from subverting an agent's reputation by becoming score-managers for agent a.

In recent work [5] we showed that reputation-based trust management can be leveraged to enforce strong data integrity policies in P2P networks. A Penny network enforces integrity policies by associating global trust labels with both agents and shared objects. A trust label t_o associated with an object o serves as a *global integrity label* for the object. The list of objects returned by a search query submitted to a Penny network includes each object's global trust label as well as the global trust labels of any servents from which the object can be downloaded. Thus, Penny users can decide whether to download an object based on its global integrity label, and they can decide from whom to download the object based on each peer's global trust label. After downloading object o from servent a, the downloading agent can report feedback to each of object o's score-managers as well as to servent a's score-managers, thereby updating t_o and t_a.

Equipping a P2P network with a trust-management system greatly increases the preventative power of intrusion detection systems such as honeypots. Specifically, it permits such systems to have an immediate and global impact on malware propagation throughout the network. Since object labels are global, malware in a P2P network equipped with a trust management system quickly accrues a poor reputation once a reputable agent detects it and reports feedback for it. A honeypot that detects and reports malware regularly will accrue a very high reputation because its numerous opinions will be independently corroborated by a large and diverse collection of non-malicious agents. When non-malicious agents outnumber malicious agents, honeypots become more influential than malicious collectives even if malicious agents outnumber honeypots. As a result, objects reported as malware by honeypots incur an immediate and drastic drop in their global reputations, thereby warning potential downloaders and inhibiting malware propagation.

C. Secure Routing

Trust management technologies are only an effective means of enforcing integrity policies when agents can successfully contact score-managers to retrieve accurate global reputations for objects and peers. This introduces a problematic circularity: to route messages securely one must avoid routing them through low-trust peers, but to identify low-trust peers one must securely route messages to and from score-managers.

Trust-management systems alone are therefore not enough to develop a secure P2P network; the trust management system must be implemented atop a secure routing protocol. Secure routing in P2P networks remains a difficult problem, but in this section we describe various promising research directions as well as important open problems concerning this subject.

Attacks upon the routing structure of a P2P network come in at least four forms:

- Malicious agents might silently drop messages that they should forward.
- Malicious agents might misroute messages to delay or prevent delivery [15].
- A malicious agent might lie about its placement in the overlay topology, causing the routing tables of other agents to be corrupted and causing non-malicious agents to misroute messages to the malicious agent.
- In a *Sybil attack* [16], a malicious agent masquerades as many different agents in an effort to control a large percentage of the identifier space and cast many votes.

In the case of the first form of attack, P2P network protocols like Chord [7], CAN [10], Pastry [11], and Tapestry [12] all have built-in fault-tolerance that adapts the routing structure to agent failures, but they do not protect against agents that maliciously drop messages. For example, Tapestry networks route messages around failed nodes that do not respond to a periodic probe message, but they do not detect or circumvent malicious nodes that respond to probes yet drop other messages.

One approach toward addressing this problem is to add non-determinism to the routing protocol. In most P2P network topologies there exist many possible routes from one peer to another, even though a deterministic routing protocol will always choose the same one for any given pair of peers.[1] Adding non-determinism increases the chances that repeated attempts to send a message will eventually circumvent malicious nodes and result in successful delivery. In a non-deterministic protocol the route chosen will not always be the best route available, so preserving the efficiency of network operations requires a strategy for probabilistically choosing amongst the available routes in a way that balances the expected success rate against the expected delay in delivery.

Malicious peers that misroute messages instead of dropping them can effect a different form of denial of service—an *attrition attack* [17]. In this attack, the malicious peer misroutes messages in such a way that other peers waste bandwidth and other computational resources attempting to deliver the misrouted message. When the malicious peer can cause a disproportionally large amount of waste relative to the attacker cost the attack is more effective than typical network-level flood attacks.

One promising defense against such attacks involves combining *self-certifying identifiers* [18], [19] with *constrained routing tables* [20]. A peer's position in a P2P overlay is determined by a unique identifier assigned to the peer, usually derived by applying a secure hash function to the peer's IP number. Self-certifying identifiers extend these identifiers with the bits of a public key in an asymmetric key pair. This allows a peer to prove that it owns a given identifier by signing its responses with the private key of the pair. Once a peer can verify the identifiers of peers with whom it communicates, it can constrain its routing table to reject messages that have been routed too far off course. The receiving peer only forwards the message if one of the acceptable routes from the sender to the destination includes the receiving peer. This limits the degree to which malicious peers can misroute a message because routing a message away from the intended target will cause the message to be rejected and the malicious peer will suffer a drop in reputation.

Perhaps the most difficult form of attack faced by P2P networks is the Sybil attack [16]. In this attack, a malicious agent that controls a large pool of IP addresses joins the P2P network many times using a different IP address each time. This allows her to control a large portion of the identifier space, which can increase her voting power and improve the odds that she can occupy all routes between a given pair of endpoints, facilitating denial of service attacks.

The best protection against Sybil attacks currently comes in the form of cryptographic puzzles (c.f., [21]). In this defense, newcomers to a P2P network are required to solve a randomly

[1]Routes can change as peers join and leave the network, but such changes to the overlay structure are localized so that the probability that they will affect any given route tends to be small.

generated mathematical puzzle in order to obtain a network identifier. The puzzle is chosen so that it is tractible for a typical end-user machine, but solving hundreds or thousands of instances of the puzzle would be computationally prohibitive. Attaching a computational cost to obtaining a P2P network identifier makes it difficult for most attackers to acquire too many identifiers. Unfortunately it can be difficult to assign a cost that is prohibitive to attackers with many computational resources but not prohibitive to loyal nodes that may have fewer resources.

An intriguing alternative is to track the history of which peers induct which other peers into the network [22]. A Sybil attack typically begins with one malicious node convincing a non-malicious node to induct her into the network. Once the malicious node has joined, she can induct her other aliases into the network directly without convincing the non-malicious node again. These self-inducted collections of malicious nodes can therefore be detected by looking for large collections of low-reputation peers all of whom have been inducted into the network by the same peer. Evicting such collections from the network would force attackers to resort to distributing their Sybil attacks over a wider collection of non-malicious peers, which is both more difficult to accomplish and provides more opportunities for non-malicious peers to detect and respond to the attack.

With protection against malicious message-dropping, attrition attacks, identifier forgery, and sybil attacks, P2P networks can withstand an impressive array of availability attacks. In the next section we argue that this secure infrastructure can be leveraged to enforce useful access control policies for secure data sharing.

D. From Trust Labels to Access Control

We have already argued that with a reputation-based trust management system implemented atop a secure routing protocol, one can enforce strong data integrity policies in a distributed setting. Extending this to enforce access-control policies is non-trivial. In this section we highlight some of the subtleties involved.

The reputation-based trust management systems discussed in Section II-B maintain a global integrity label for each object in the system. This can easily be extended to a vector of labels based on different criteria—e.g., integrity labels and confidentiality labels. Combined with global trust labels for peers, this permits the enforcement of discretionary access control policies where peers are subjects. For example, before servicing a download request, a peer can consult the global security labels for the requested object and the global trust label of the requesting peer. If the integrity label of the object is too low, or the confidentiality label of the requested object is too high relative to the trust label of the requester, then the peer refuses the request. This prevents the spread of low-integrity data and prevents low-trust peers from obtaining high-confidentiality data.

However, while the above strategy suffices to enforce discretionary read-access policies based on data integrity, it misses an important subtlety related to enforcing confidentiality policies. In order for a trust management system to enforce any security policy, violations of the policy must get reported so that violator reputations will be downgraded. In this way future violations are prevented. Although there are many scenarios wherein integrity violations are reported (e.g., a non-malicious peer downloads a file and discovers that its content is not what was requested), it is not clear how confidentiality violations ever get reported. Confidentiality violations typically involve one malicious peer divulging confidential data to another malicious peer, in which case neither peer is likely to report the violation.

Honeypots can potentially detect confidentiality violations, but making productive use of this information within the trust management system can be problematic. For example, a honeypot might randomly request high-confidentiality objects from other peers in an effort to detect information leaks. If the honeypot maintains a poor global reputation, then any peers that service its requests are guilty of confidentiality violations and will be reported by the honeypot. Unfortunately, since the honeypot must maintain a poor reputation in order to test for confidentiality violations, its reports of violators will carry little weight in the trust management system.

Confidentiality policy enforcement therefore remains a difficult open problem in P2P networks. Trusted computing platforms might be the only solution at present, since they allow global security policies such as mandatory access control policies to be enforced remotely [23]. A P2P network based on trusted computing would verify that each peer is running trusted hardware and software before admitting it to the network. Trusted hardware and software would be required to obey the P2P network protocol and serve data in accordance with the system-wide access control policy. While this strategy might become feasible as trusted computing architectures become more widely available, it remains inappropriate for P2P settings where users desire greater control over their own client systems. In what follows we therefore limit our attention to access control policies based on data integrity rather than confidentiality.

III. Secure Data Management

Secure P2P networks and trust-based reputation systems provide a means of enforcing important low-level access control policies such as role-based access control and integrity policies. Our challenge is to develop trusted applications that could be hosted atop such an infrastructure. To explain the issues involved we will consider data management applications.

In a data management system, access to the data can be controlled based on association/context as well as content. Therefore, the policies are richer than those developed for networks and operating systems. With respect to integrity and trust in data management systems, the challenges include:

- To what extent does one trust the data?
- Is the data accurate?
- How can one maintain data provenance so that data misuse can be detected?

- How can we compute trust values to associations between data? For example, if the trust value for data object A is t_A and the trust value for object B is t_B, then what is the trust value of the fused data object (A, B)?

In this section we consider some of the challenges that must be investigated in order to answer these questions.

A data manager essentially manages a collection of database objects. These objects can be viewed using various data models including relational models and object models among others. There has been extensive research in the past on hosting secure data managers on secure operating systems (c.f., [24]). For example, various multilevel secure data management systems have been designed and developed. These systems typically enforce the Bell and LaPadula security policy [25] where database objects are assigned sensitivity levels and the users are assigned clearance levels. User access to the database is controlled by the simple property and the star property.

A major challenge in designing such a multilevel data management system is the granularity of classification. In operating systems the files are assigned sensitivity levels, but in data management systems object sensitivity levels might depend on the content, context, and time. For example, a document published by the CIA could be highly classified while a document published by a university could be unclassified.

In addition, some database objects are a fusion of other objects. For example, consider an English document produced by authors from multiple countries. The chapters written by native English speakers might have higher integrity values than those written by non-English-speaking authors. In this situation, which integrity value does one assign to the book? Should it be the lowest integrity value amongst all of the integrity values of the chapters? Should it be a higher value that is the average of all the chapter integrity values?

Another important issue regards how to represent these fused objects at the network level. For example, consider representing a book in a data management system built atop a Penny [5] network. Users might want to download individual book chapters or the book as a whole. Thus, one strategy would be to represent each chapter as an individual Penny object and the entire book as a separate Penny object. However, this scheme introduces a prohibitively expensive storage cost in the worst case. That is, we might need a separate Penny object for each subset of book chapters, causing the storage costs to rise exponentially with the number of books.

Alternatively, one might represent the book as a Penny object that consists of a collection of pointers to chapter objects. The challenge then is to design a method for assigning and tracking trust levels for these composite objects as the trust levels of their constituent objects change. Questions also arise regarding how to interpret feedback reported for one of these composite objects. If a Penny peer reports an integrity label for a book object, does that integrity label get applied to all the chapter objects, or does it indicate the integrity of the pointers themselves but not the integrity of the objects they point to?

Containment is only one of many relationships that might exist between objects at the data management layer. For example, many modern data management systems are based on the relational model. These relational database systems are being used for a variety of applications including e-commerce, situation awareness, and intelligence analysis. The granularity of classification for a relation could be at the table level, at the attribute level, or at the element level. Furthermore, relational operations such as the join operation results in new relations with security labels possibly derived from the underlying relations that were joined.

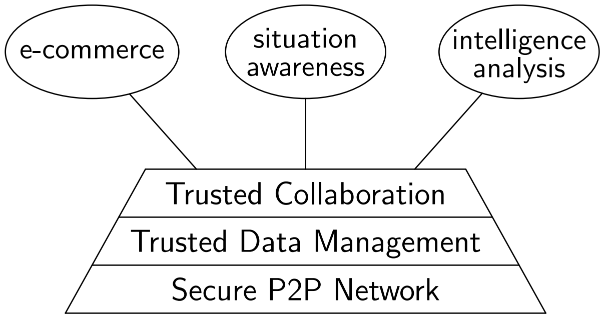

Fig. 1. A layered approach to trusted collaboration

In the case of confidentiality labels, a challenge would be to ensure that the higher-classified data cannot be inferred from lower-classified, joined data. In many cases the joined data might divulge associations and so might need to be assigned a higher security label. Such security properties must either be enforced at the secure P2P network layer or we need policy extensions so that the applications built atop this layer enforce the additional policies. Once the security labels of the elements of the joined relation are determined, then they could be decomposed into atomic P2P network objects and access controlled by the secure P2P networking protocol.

IV. Trusted Collaboration

Secure P2P networks and trusted data management can form the basis for trusted collaboration at a high level. The idea here is for different organizations to share data and carry out collaboration and decision-making. Figure 1 illustrates a layered approach to trusted collaboration that utilizes the secure P2P infrastructure and trusted data management systems described in the previous sections. The data management and network layer should provide appropriate services to ensure trusted collaboration.

The challenges here include the trust that an organization places on another organization. Note that the 9/11 commission report states that we need to migrate from a need-to-know environment to a need-to-share environment. Furthermore, to fight the global war on terror, we need to work with trusted, untrusted, and semitrusted partners. Therefore, we need to answer questions such as:

- Do we share data when requested and then determine the consequences?
- Do we share only partial data with partners who are not entirely trustworthy?

- If so, how do we determine the data that needs to be shared?
- How can data-sharing be supported in both push and pull models?
- Do we share data based on risk?
- What are the challenges in developing a risk-based trust model for data sharing?

These are some of the research challenges that need to be investigated for trust collaboration to be built atop the secure infrastructure.

V. Conclusion

The past five years have seen numerous significant advances toward enforcing important security policies in P2P networks. A particularly active and challenging area of research involves developing secure P2P network routing protocols that enforce availability policies in the presence of malicious peers. Malicious peers might drop messages, misroute them, or otherwise disrupt normal traffic in the network by violating the networking protocol. Many of these attacks can be thwarted by employing technologies such as self-certifying identifiers, constrained routing tables, probabilistic routing protocols, and cryptographic puzzles.

We have shown that these low-level security enforcement mechanisms can be used as a foundation for enforcing certain higher-level data security policies. In particular, we show how to use a reputation-based trust management system to enforce discretionary access control based on global integrity labels. Other policies such as confidentiality enforcement and mandatory access control remain open problems but are a subject of active research on emerging technologies such as trusted computing platforms.

These advances seem poised to support next-generation secure applications for trusted collaboration atop P2P networks. We considered the challenges involved in implementing such systems, and we highlighted outstanding open problems. These include issues related to confidentiality policy enforcement and the need to reflect security policies and security labels at the data management level down to the level of the P2P network object infrastructure. Despite the challenges, we argued that the rapid maturing of P2P security research has brought solutions to these issues within reach, and we have advanced strategies for tackling these important problems.

Acknowledgment

The authors would like to thank Nathalie Tsybulnik for many fruitful discussions about P2P network security and her helpful critiques of this paper.

References

[1] Napster, http://www.napster.com.

[2] Gnutella, http://www.gnutella.com.

[3] D. S. Wallach, "A survey of peer-to-peer security issues," in *Software Security—Theories and Systems, Mext-NSF-JSPS Int. Symposium, ISSS*, Tokyo, Japan, November 2002, pp. 42–57.

[4] J. Risson and T. Moors, "Survey of research towards robust peer-to-peer networks: Search methods," *Computer Networks*, vol. 50, pp. 3485–3521, 2006.

[5] N. Tsybulnik, K. W. Hamlen, and B. Thuraisingham, "Centralized security labels in decentralized P2P networks," in *Proc. Annual Computer Security Applications Conf. (ACSAC'07)*, Miami Beach, Florida, December 2007, to appear.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proc. 12th Int. World Wide Web Conf. (WWW'03)*, Budapest, Hungary, May 2003, pp. 640–651.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. ACM Conf. on Applications, Technologies, Architectures, and Protocols for Comp. Comm. (SIGCOMM'01)*, San Diego, California, August 2001, pp. 149–160.

[8] KaZaA, http://www.kazaa.com.

[9] Limewire, http://www.limewire.com.

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable, content-addressable network," in *Proc. ACM Conf. on Applications, Technologies, Architectures, and Protocols for Comp. Comm. (SIGCOMM'01)*, San Diego, California, August 2001, pp. 161–172.

[11] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Int. Conf. on Distributed Sys. Platforms (Middleware '01)*, Heidelberg, Germany, November 2001, pp. 329–350.

[12] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE J. on Selected Areas in Comm. (JSAC'04)*, vol. 22, no. 1, pp. 41–53, January 2004.

[13] S. Shin, J. Jung, and H. Balakrishnan, "Malware prevalence in the KaZaA file-sharing network," in *Proc. 6th ACM SIGCOMM Internet Measurement Conf. (IMC'06)*, Rio de Janeiro, Brazil, October 2006, pp. 333–338.

[14] A. Kalafut, A. Acharya, and M. Gupta, "A study of malware in peer-to-peer networks," in *Proc. 6th ACM SIGCOMM Internet Measurement Conf. (IMC'06)*, Rio de Janeiro, Brazil, October 2006, pp. 327–332.

[15] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Proc. 1st Int. Conf. on Peer-to-peer Sys. (IPTPS'02)*, Cambridge, Massachusetts, March 2002, pp. 261–269.

[16] J. R. Douceur, "The Sybil attack," in *Proc. 1st Int. Workshop on Peer-to-peer Sys. (IPTPS'02)*, Cambridge, MA, March 2002, pp. 251–260.

[17] T. J. Giuli, P. Maniatis, M. Baker, D. S. H. Rosenthal, and M. Roussopoulos, "Attrition defenses for a peer-to-peer digital preservation system," in *Proc. USENIX Annual Technical Conf.*, Anaheim, California, April 2005, pp. 163–178.

[18] J. Hautakorpi and J. Koskela, "Utilizing HIP (host identity protocol) for P2PSIP (peer-to-peer session initiation protocol)," Internet-Draft draft-hautakorpi-p2psip-with-hip-00 P2PSIP WG, July 2007, (http://tools.ietf.org/wg/hip/draft-hautakorpi-p2psip-with-hip-00.txt).

[19] T. Aura, A. Nagarajan, and A. Gurtov, "Analysis of the HIP base exchange protocol," in *Proc. 10th Australasian Conf. on Info. Sec. and Privacy (ACISP'05)*, Brisbane, Australia, July 2005, pp. 481–493.

[20] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *Proc. 5th Symposium on Op. Sys. Design and Implementation (OSDI'02)*, Boston, Massachusetts, December 2002.

[21] S. Ryu, K. Butler, P. Traynor, and P. D. McDaniel, "Leveraging identity-based cryptography for node ID assignment in structured P2P systems," in *Proc. 21st Int. Conf. on Advanced Information Networking and Applications (AINA'07)*, Niagara Falls, Canada, May 2007, pp. 519–524.

[22] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson, "Sybil-resistant DHT routing," in *Proc. 10th European Symposium on Research in Comp. Sec.*, Milan, Italy, September 2005, pp. 305–318.

[23] S. Balfe, A. D. Lakhani, and K. G. Paterson, "Trusted computing: Providing security for peer-to-peer networks," in *Proc. 5th Int. Conf. on Peer-to-peer Computing (P2P'05)*, Konstanz, Germany, August 2005, pp. 117–124.

[24] B. Thuraisingham, *Database and Applications Security: Integrating Information Security and Data Management*. Boca Raton, Florida: Auerbach Publications, 2005.

[25] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," The MITRE Corporation, Bedford, Massachusetts, Tech. Rep. MTR-2547, Vol. I, ESD-TR-73-278-I, March 1973.