Avoiding Eclipse attacks on Kad/Kademlia: an identity based approach

Leonardo Maccari*, Matteo Rosi*, Romano Fantacci*, Luigi Chisci ‡ Luca Maria Aiello†, Marco Milanesio†
*Department of Electronics and Telecommunications - University of Florence
Email: {maccari, rosi, fantacci}@lart.det.unifi.it
*Dipartimento di Sistemi e Informatica - University of Florence
Email: chisci@dsi.unifi.it
† Department of Computer Science - University of Turin
C.so Svizzera 185, 10149 Torino, Italy
Email: aiello.luca_maria@educ.di.unito.it, milane@di.unito.it

Abstract—**Kademlia is a Distributed hash table widely used in P2P networks that has been applied to commercial and non commercial distribution of files. In this paper the authors review some security issues connected with Kademlia and propose a novel technique to leverage its security using an external certification service.**

I. Introduction

Peer to peer (p2p) networks are distributed networks formed by software applications that create an overlay on the Internet. Many models of p2p networks have been proposed in theory, some of them have been implemented but only a few are used by a large community. The main advantage over a centralized network is the distribution of the load necessary to store and to distribute data but their complexity, compared to a centralized network is much greater.

Existing networks are mainly targeted at file-sharing between users; each user decides which resources to share and publishes in the network an index associated with the file together with its IP address. While this model makes the network robust and eliminates single points of failure, it makes it impossible to control the behaviour of the users. For this reason p2p networks have been the target of the complaints of the media industry, claiming that copyrighted contents are exchanged without appropriate licenses.

Nevertheless, p2p networks have gained great attention because they are a possible solution to the problem of distributing data without the effort needed for a centralized server. Examples of commercial networks based on the p2p model already exist for file-sharing [1], voice over IP [2] and video streaming [3].

In recent years, P2P systems took advantage of the adoption of Distributed Hash Tables (DHT), a set of distributed algorithms that, partitioning the ownership of a set of keys among participating nodes, provide scalability and robustness to the applications built on them. This paper concentrates on the robustness of Kademlia DHT. Kademlia [4] has been implemented in many software suites such as Emule/Kad, Overnet and Azureus, currently used by millions of users everyday. This makes a Kademlia-based network an interesting subject for an attacker, that could be interested into spying user actions, substituting resources with fake ones, or subverting the protocol in order to produce a denial of service attack against a third party. In this paper the security of Kademlia will be analyzed and a possible solution will be proposed.

The rest of the paper is made up of two parts. In the first, the security of Kad (and consequently Kademlia) will be analyzed through computer simulation with special attention to eclipse attacks; in the second a possible solution will be proposed. The proposed solution is based on the adoption of a certification service and of an authentication protocol between nodes. This will allow to introduce credentials in order to make the ownership of the contents and messages inserted in the DHT non-repudiable.

II. Overview of Kad/Kademlia

Distributed Hash Tables have grown in popularity and reached a lot of different application domains, enriching the applications built on them with scalability, robustness, consistency and efficiency features. A DHT is a hash table which contains pairs of the type *(key,value)* that can be distributed over an arbitrary amount of users. In this is used in p2p networks to distribute the associations between a key that identifies a certain resource and its *value*, namely the IP address of the node that contains it.

In Kademlia, each node of the network has an identifier (ID) consisting of a 160 bit vector generated using the SHA-1 hash function on a random value. The nodes of the network can then be represented as leaves of a binary tree with 160 levels. Clearly, the number of nodes is much lower then 2^{160}, so it turns out that the majority of the leaves correspond to free IDs. For this reason, each node can be identified by the initial portion of its ID that is distinct from any other ID present in the network.

A distance between two IDs is defined as the XOR between them. The distance intuitively represents the number of levels that separate two leaves when navigating from one node to another. The order of magnitude (base 2) of the distance

This work is partially supported by the Italian MIUR project PRIN 2006 *PROFILES* (Peer to peeR beyOnd FILE Sharing)

expresses the number of levels that must be navigated up in the tree to go from one node to another. In this way each node can maintain a set of *buckets*, i.e., ordered lists of nodes it has been in contact with in the past. Each bucket contains IDs that have the same distance (in order of magnitude) compared to the ID of the node itself.

When entering the, a node will have just empty buckets apart from one point of attachment to the network. During its lifetime it will enter in contact with other nodes and will slowly populate its buckets.

Kademlia supports various *Remote Procedure Calls* (RPC) implemented as UDP frames with an appropriate opcode. Two specific RPC that will be considered in this work are the FIND_NODE() and the FIND_VALUE(), they both take as a parameter a target ID. When using the first one, the receiver will answer with a set of IDs closest to the target taken from its buckets (including also the corresponding IP address and UDP ports). When using the second one, the receiver will answer, if it owns it, with the resource associated with the target.

Routing (i.e., the procedure of resolving an ID of a node into its IP address) is realized by sending iterative FIND_NODE() requests. When looking for a certain ID, a node will send three FIND_NODE() to the closest nodes it knows, then for each response it will send three more requests to the three closest new nodes and so on.

It is easy to see that if each node has at least one active node in each of its buckets and each node knows all its closest neighbours (which are quite realistic assumptions), for any hop the distance is reduced by a half, so that the routing algorithm will converge in $log_2(distance)$ steps to the ID closer to the target. The search process ends when the target is reached, or when issuing new requests does not return any node closer than the known ones.

The resources (i.e., files) are distributed over the same space and routing is used also to inject a resource into the network. If node A wants to share a file x, it calculates ID_x = `SHA-1(x)` and performs multiple actions: first of all a routing request is issued for ID_x, and will end to the alive node closest to ID_x denoted by B. B will receive a specific RPC, STORE(ID_x, IP_A). Then B, the closest node to ID_x, is responsible for this piece of the index.

When a user wants to search for the file x in the network, she must know ID_x and she will issue a FIND_NODE(ID_x). The RPC will end in node B. Then the user will issue a FIND_VALUE(ID_x) to node B and node B will answer with the IP address of node A. The user then can open a TCP connection to node A and start downloading the file.

A. The Kad implementation

The most successful implementation of the Kademlia DHT is the Kad network, that has been used in the eMule p2p software. As observed in [5], Kad is populated by a number of users between 3 and 5 millions spread around the world and contains almost 80 millions shared resources. The Kad network exhibits some differences with respect to the original Kademlia DHT; in particular, it introduces the concept of a *tolerance zone*. Whenever a node A wants to share a resource x, it issues a routing request (actually multiple routing requests in parallel) to find the node responsible for ID_x. During the iterative request, node A collects IDs of other nodes and performs the STORE() operation on the first 10 nodes it encounters that have at least the 8 most significant bits in common with ID_x. An 8-bit zone is called the *tolerance zone* for ID_x. As a consequence, the search operation is different. If node B wants to retrieve resource x, it issues an iterative search for ID_x; whenever it receives the IP of a node that falls into the tolerance zone, it issues a FIND_VALUE(ID_x) for ID_x to that node; in response, a list of possible resources is returned (for example, a list of IP addresses that contain the resource). The iterative search for ID_x and the FIND_VALUE(ID_x) for ID_x keep going in parallel until:

- a timeout is fired,
- the routing to ID_x expires because no new nodes are found,
- a maximum number of resources are returned by the FIND_VALUE() primitives. This value depends on the kind of resource being searched.

Kad contains many distinct kinds of resources, among which there are keywords and files. Keywords are used to reconstruct the ID of a file: when a user wants to share a file x, he calculates the hash of the file ID_x, as well as the hash of each of the words that compose the file-name, and stores the association (ID_{word}, (ID_x, *"complete title"*)) in the network. When a user wants to search for a file, it issues separate searches for the hashes of all the words that are in the file-name. From the intersection of the responses the user will select the most appropriate one depending on the title and start a look-up for ID_x, which will return a list of IP addresses that contain that file. Then, the node can directly contact the given IP addresses to download the file.

Whenever a node shares a file x with ID_x and a certain file-name, it is responsible for periodic republication of the IDs of the file and of the title-words, each republication being performed on 10 nodes in the corresponding tolerance zone. This means that if a file is shared by many users, its ID is republished frequently, and a higher number of nodes in the corresponding tolerance zone stores the IP of the owners. When a node receives a store message, it will start a timeout, and after a fixed amount of time, the resource is erased if no update is received. The expiry time and the republication time depend on the kind of resource, varying from 5 hours for keywords to 24 hours for files. Lastly, Kad network uses an hash space of 128 bits instead of 160.

Each node periodically sends PING packets to the nodes in its buckets and purges the ones that do not respond, so that nodes with longer lifetime are the last to be removed. Since a new contact is added only to a non-full bucket, it is not trivial to perform *bucket poisoning*, i.e. send unsolicited pings to a certain node in order to be present in its bucket.

For the sake of brevity, many details of the Kademlia protocol and Kad implementation have been omitted; the interested reader is referred to [4], [6]. From now on, for clarity purposes, with the notation ID_x we will refer to the node that is identified by the ID_x, and the context will be clear enough to understand the difference between nodes and IDs. When referring to the Kad network, the superscript f will be used for IDs that refer to files (i.e ID_x^f) and k for IDs that refer to keywords (i.e. ID_x^k)

III. Simulations on the Eclipse Attack

To fully understand the security issues that affect the Kademlia DHT, a simulation-based analysis has been performed focusing on the evaluation of the eclipse attacks. The simulations are realized with the Omnet++ simulator on a single-layer Kad network. By means of simulations can be better evaluated the efectiveness and the consequences if such an attack, to design adequate countermeasures. Two kinds of attacks will be analyzed, i.e. the pure eclipse attack and a variation with randomly chosen nodes. It will be shown that the impact of the attack depends mainly on three factors: the number of attacking nodes, the possibility of the attacker to choose IDs arbitrarily and the possibility to start the attack before the resource is published.

A. The Eclipse Attack

The eclipse attack [7] is a form of routing poisoning which aims to separate a set of victim nodes from the rest of the overlay network. A not limited set of attacker nodes reaches its purpose by fooling correct nodes into adopting malicious nodes as their peers, with the goal of entirely dominating their neighbor set. If carried out successfully, the eclipse attack allows an attacker to mediate most overlay traffic and effectively eclipse correct nodes from each other's view.

Briefly, the eclipse attack is performed by an attacker that tries to intercept all the requests directed to a specific resource and redirect them to a fake resource. It has already been shown that this attack is possible in Kad if the attacker controls a set of IDs close to the target ID (i.e. ID_e) before the publication of ID_e, see [8].

The eclipse attack can be also intended as an attack against the storage; when it's targeted to overshadow content stored on DHT, making them inaccessible to lookup requests, it takes the name of *node insertion* attack. It is carried into effect by initiating a substantial number of nodes marked with identifiers numerically close to the ID_e, so that all lookup requests for k are routed to these nodes. Once received a lookup request they can answer with a fake content or they can send no reply message at all, effectively hiding the content to the DHT. Node insertion attack cannot be fought with anonymous auditing.

It should be noted that the various manifestations of eclipse attack can effectively take place only if the attacker nodes may select their node id "ad-hoc", so they can populate the target nodes' route table entries or the keyspace depending on whichever is the key associated to the node or to the content they aim to eclipse.

B. Simulation Results

Simulations have been performed observing the behaviour of a single tolerance zone containing 4,000 nodes, corresponding to a network composed of approximately 1,000,000 nodes. To lower computational load, not all the network was equally popoulated. Lifetime of the nodes and other system parameters were chosen consistently with the measurements done in [5], [6] and following works by the same authors. The deviation of measured data between two simulations with distinct random seeds is extremely low (below 2%) so in the figures are reported the profiles given by an average one. After a set-up phase, the attacks start at second 10,000.

The simulations confirmed what observed in [8], that if the attacker is able to choose the IDs arbitrarily and place those IDs in the network before ID_e is published, the attack reports almost 100% of success. This means that almost all the requests for ID_e are captured by the attacker; to achieve this, the attacker needs to control just 8 IDs. In Kad, users choose their own IDs. If ID_e is an ID of a keyword, then it is predictable and can be eclipsed prior to its publication.

Through simulations it is possible to estimate the impact of eclipse attacks even in the case when ID_e is a file hash. In that case, it is not known before its publication and the attack starts after it has been published. This means that there is a race condition between the attacker spreading fake resources and the other nodes spreading correct ones. In this scenario with the same 8 attacking IDs the attack doesn't have a high impact if not supported by a bucket poisoning strategy. With such a strategy, the attacker IDs send unsolicited pings to the nodes in the tolerance zone of ID_e so that such IDs will be added to the buckets of correctly behaving nodes. This will make it easier for the attacker to be contacted. In figure 1 the eclipse attack is started by 8 attacking nodes after the publication of ID_e together with bucket poisoning. It can be noticed that as time passes the attack becomes more effective, but even on a longer time span it doesn't reach a total eclipse. This is explained by the fact that for every obtained request, the resource is republished by the receiver on ten nodes in the tolerance zone, as it would happen for a downloaded file in Kad. There is a concurrency between the republication of correct resources and fake ones, that prevents the attack from being completely successful.

A variation of the eclipse attack has been considered when the attacker is not able to determine its own ID. In [9] and [10] it is suggested that IDs could be chosen as a hash of network parameters (IP address and UDP port). Our simulations show that even in that case, if the attacker owns a large number of IDs, the eclipse can still happen with high probability. With a single IP address, a node can obtain 2^{16} different IDs uniformly spread. This yields about 250 IDs per tolerance zone. In figure 2 it can be seen that even when the attack start after the publication of ID_e, with bucket poisoning, the percentage of eclipsed requests is still high. Moreover in this scenario there is no concentration over a single ID so that attacker could eclipse *any* resource in the tolerance zone. This

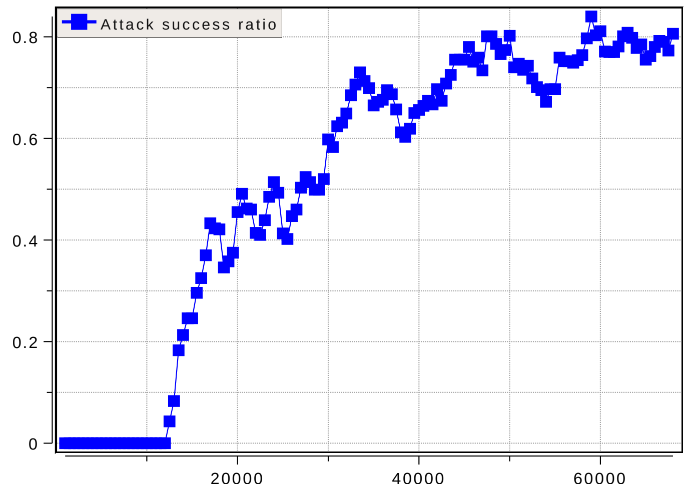

Figure 1. The ratio between the number of search requests for ID_e and the number of search requests that are intercepted by an attacker node. Measures are sampled every 500s and averaged over a sliding window of 10 samples.

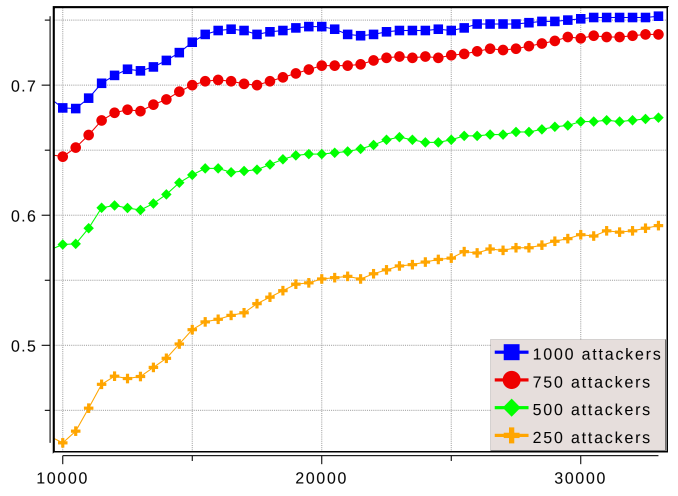

Figure 2. The ratio between total number of lookups in the tolerance zone and lookups intercepted by the attacker IDs

means that the attacker could perform a monitoring activity, substitution of resources or DoS against any resource in the tolerance zone with the success ratio shown in figure 2.

The last attack that has been evaluated through simulations, is a DDoS attack against a third party. This last scenario can be used to produce a different attack. As explained in [11] if all the attacking IDs answer to FIND_VALUE() RPCs with the IP of a victim node (even outside of Kad), associated to fake IDs, this node will be flooded by requests coming from multiple sources. This will produce a Distributed Denial of Service extremely hard to stop. From our simulations we were able to estimate that with FIND_VALUE() RPCs the attacker receives an average of 4.5 frames for each frame sent by the attacker. This means that the upload bandwidth resources of the attacker are multiplied by 4.5.

From the simulations emerges that, qualitatively, Kad is vulnerable to various attacks that are generated by two factors: the possibility for the attacker to choose its own IDs and the possibility to have multiple IDs. In the following part of the paper it will be shown how the introduction of a certification service can help Kademlia (and consequently Kad) to be more resistant against the attacks that exploit these factors.

IV. Proposed Countermeasures

As underlined before, one of the starting point for the eclipse attack to a DHT based system, is the possibility for an attacker, to compute its own identifier, or at least, to join the network with many randomly chosen identifiers. In this section, we present a possible way to prevent the attackers from choosing their own identifiers and from easily instantiating a large number of nodes with the introduction of a Certification Service that signs all the valid identifiers in the network. Furthermore, we introduce a novel node interaction protocol based on the exchange of signed tokens exchange that protects the system from other well known attacks.

In Kademlia, nodes' identifiers are not certified and they can be generated at will on the local node, so that it's possible to quickly instantiate a large number of Sybil [12] nodes with arbitrary IDs in order to complete an eclipse attack. There is no credential associated with contents maintained in storage areas and no control is performed by replica nodes over the information stored in the DHT thus allowing the index poisoning [13] and derivative attacks like DDOS [11]. There is no authentication protocol between nodes. Nevertheless, k-buckets provide resistance to routing poisoning attacks; in fact, one cannot flush nodes routing state by flooding the system with new nodes. Kademlia nodes will only insert the new nodes in the k-buckets when old nodes leave the system. Unfortunately, it is very easy to inject into a route table information relating to contacts whose identifier is very close to the victim node ID, because of the bucket splitting procedure.

Our framework [14] includes an identity based scheme and a secure communication protocol that may provide an effective defense against well known attacks. The proposed approach is layered on Kademlia and its architecture is based on the presence of a Certification Service (CS). The CS can be a centralized or decentralized authority whose task is to generate random node IDs and to certify the links between node IDs and users' identities by signing peculiar tokens. To accomplish this, we suppose that a classic public key cryptography scheme is used: in this section it is assumed that the CS is a centralized authority owner of a public key known to every Kademlia node, and holder of its private counterpart. Similarly, it is assumed that each user should be in possession of a key pair. In the following, a brief explanation of the protocol is given. For more details, please refer to [14].

Before the bootstrap, a node needs to send a proper $NodeIdRequest$ to the Certification Service in order to retrieve its own certified identifier, querying the service with its user's identifier (e.g., an email address or an OpenID) and public key. The CS returns a signed identifier, the $AuthId$, containing the node ID, the user ID and the public key.

The CS makes the user identifier verification procedure (whose steps depend on the nature of the identifier itself), and then binds the user identity with his public key and with a $NodeId$ by producing the following token:

$$AuthId_A = Sign(NodeId_A||UserId_A||K_A^+||exp_A, K_{CS}^-)$$

The $NodeId$ is randomly chosen; exp_A is a timestamp that establishes the expiration date of the signed $NodeId_A$. This token is signed with the private key of the CS. The CS keeps track of the association between $UserId$ and $AuthId$, so that all subsequent $NodeIdReq$ received by the same users receive in response the same $AuthId$ passed earlier, unless it is expired or close to expiration. This is a precaution to prevent the CS from producing useless signatures. The CS also returns a set of bootstrap nodes, for the new node to join the system.

A node A can then successfully send a RPC (join primitive included) to a node B and obtain a proper response only if both A and B observe the following communication protocol:

I $A \rightarrow B : NodeId_A, N1$
II $B \rightarrow A : NodeId_B, N2$
III $A \rightarrow B : AuthId_A, Auth_{AB}$, RPC-REQ
IV $B \rightarrow A : AuthId_B, Auth_{BA}$, RPC-RES

We call this four way exchange a *session* between A and B. RPC-REQ and RPC-RES fields are respectively the request and response RPC defined in Kademlia; $N1$ and $N2$ are randomly generated nonces, used against man-in-the-middle attacks. Messages sent at steps I and II must be somehow marked differently (e.g. different opcode), to distinguish the request from the response. A generic $Auth_{XY}$ is a signed token containing X's node ID, a hash code of the RPC and the nonce previously received from Y.
The above described interaction affects also the content storage system of the single nodes. Each resource contained in a STORE RPC is associated with some signed *credentials*, binding the user identifier to the key for which the content was inserted and to the hash code of the content, so that it it possible to verify that the owner had inserted a certain content with a certain key.

A. Discussion

In this section we discuss how this proposal strongly limits the eclipse attack as well as many other possibly malicious behaviors.

In Kademlia, the sender contact of every incoming message is added to the route table if there is enough room in the buckets. The contacts with a node ID close to the local ID are always added to the route table due to the splitting procedure. To effectively put off a routing attack, the attacker must inject bad routing information in the target node by sending him messages that report sender IDs near to the victim's ID. Combined usage of $AuthId$ and $Auth$ makes the communication between nodes authenticated, so that the attacker can inject only its own contact into the target route table; since the IDs are randomly chosen by the CS, the attacker cannot generate its ID "ad hoc". Moreover, it is unfeasible for an attacker to hide a content marked with a given key k by way of a node insertion attack, because the malicious node cannot register a substantial number of nodes with IDs close to k; in fact, he cannot control ID generation by his own.

On the other side, since every user can have multiple identities (e.g. many email addresses). A user can bind each of his identities to a different node by sending many $NodeIdRequest$ to the CS, and then she can run all those nodes on the same machine. Hence the Sybil attack is not completely wiped out with this scheme. Nevertheless, each node corresponds to a different user account and the node initialization requires a verification procedure for that account. If the user authentication procedure requires a human interaction, it would be difficult for an attacker to create many different nodes in an automated way, actually lowering the risk of Sybil Attacks. For this reason, we strongly suggest to adopt OpenID verification methods, that redirect the user agent to an identity provider, and that return to the CS when the submitted identity has been correctly authenticated.

If poisoning and sybil attacks are not possible, then the eclipse attacks on the Kad network are not applicable. In fact, they were strictly dependent on the possibility of the attacker to choose arbitrary IDs or to impersonate multiple identities.

V. Conclusions

Kademlia is a DHT widely emplyed in P2P networks used by millions of users. In this paper the security of Kademlia has been studied testing the Kad implementation through simulations. As expected, the security features of Kad present various issues. The introduction of a certification service into a Distributed Hash Table can be used for denying or limiting all the attacks that can be made against a structured peer-to-peer system such as Kademlia/Kad. The authenticated identifiers are used to ensure that all the exchanged messages are non repudiable and traceable. Identifiers will be certified and randomly selected, thus preventing routing and storage attacks. In this way, the previously discussed weaknesses are strongly mitigated.

As a future work, the differences between the original Kademlia and the Kad implementation should be taken into account in order to investigate which are the new features (added in Kad) that can be exploited for an attacker. Moreover, some simulation results of the identity based approach are needed wrt the state of the node and the load on the CS: as told before, in the original approach the CS is centralized, but various solutions to the distribution of services among group of peers are under investigation.

References

[1] The bittorrent file sharing network. [Online]. Available: http://www.bittorrent.com/
[2] The skype p2p voice over ip application. [Online]. Available: http://www.skype.com/
[3] Joost, a p2p video streaming network. [Online]. Available: http://www.joost.com/

[4] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the xor metric,” *In Proceedings of IPTPS02, Cambridge, USA*, 2002. [Online]. Available: citeseer.ist.psu.edu/maymounkov02kademlia.html

[5] M. Steiner, E. Biersack, and T. Ennajjary, “Actively monitoring peers in KAD,” *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS’07)*, 2007.

[6] R. Brunner, “A performance evaluation of the kad-protocol,” Master’s thesis, Institut Eurecom, 2006.

[7] A. Singh, T. W. Ngan, P. Druschel, and D. S. Wallach, “Eclipse attacks on overlay networks: Threats and defenses,” *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–12, April 2006.

[8] M. Steiner, T. En-Najjary, and E. Biersack, “Exploiting KAD: possible uses and misuses,” *Computer communications review, Volume 37 N. 5, October 2007*, 2007.

[9] D. Cerri, A. Ghioni, S. Paraboschi, and S. Tiraboschi, “Id mapping attacks in p2p networks,” *IEEE Global Communications Conference, St. Luis, U.S.A.*, 2005.

[10] S. Balfe, A. Lakhani, K. Paterson *et al.*, “Trusted Computing: Providing Security for Peer-to-Peer Networks,” *Proceedings of Fifth IEEE International Conference on Peer-to-Peer Computing, (P2P’05), Kostanz, Germany*, pp. 117–124.

[11] X. Sun, R. Torres, and S. Rao, “Ddos attacks by subverting membership management in p2p systems,” in *NPSec 2007. 3rd IEEE Workshop on Secure Network Protocols*, 2007.

[12] J. Douceur, “The sybil attack,” 2002. [Online]. Available: citeseer.ist.psu.edu/douceur02sybil.html

[13] J. Liang, N. Naoumov, and K. W. Ross, “The index poisoning attack in p2p file sharing systems,” in *INFOCOM*, 2006.

[14] L. M. Aiello, M. Milanesio, G. Ruffo, and R. Schifanella, “Tempering Kademlia with a Robust Identity Based System,” *Eight International Conference on Peer-to-Peer Computing (P2P 2008) - to appear*, 2008.