Different perspectives of Sybil Defenses in Peer to Peer Networks

[1]Jissy Liz Jose, [2]Seena Theresa George

[1]Assistant Professor, [2]Assistant Professor
Department of Computer Science and Engineering

***Abstract*—In a peer-to-peer (P2P) network, every machine plays the role of client and server at the same time. Although a P2P network has a number of advantages over the traditional client-server model in terms of efficiency and fault-tolerance, additional security threats can be introduced. Sybil attack is one of the most challenging problems related to identity management in Peer-to-Peer networks. The huge number of fake identities created by malicious users may attempt to gain a large influence on the network and may collude or subvert the system. Here a study on different P2P system topologies, security issues in P2P networks focusing on Sybil attack and an analysis on different existing solutions to Sybil attack is done.**

***IndexTerms*—Sybil Attack, Sybil Defense**

I. Introduction

The past decade has witnessed phenomenal developments in Peer-to-Peer infrastructure, with an increase in the vast number of systems based on P2P and the research in P2P systems become an increasingly active field. P2P networks [1] use a decentralized model in which each machine, referred to as a peer, functions as a client with its own layer of server functionality. With a client-server approach, the performance of the server will deteriorate as the number of clients requesting services from the server increase. However in P2P networks, overall network performance actually improves as number of peers added to the network increases. These peers can organize themselves into ad-hoc groups as they communicate, collaborate and share bandwidth with each other to complete the tasks at hand (e.g. file sharing). Each peer can upload and download at the same time, and in a process like this, new peers can join the group while old peers leave at any time. This dynamic re-organization of peers is transparent to end-users.

Another characteristic of P2P networks is its capability in terms of fault-tolerance. When a peer goes down or is disconnected from the network, the P2P application will continue by using other peers. Compared to a client-server model, where all communication will stop if the server is down, a P2P network is more fault-tolerant. P2P computing is the sharing of computer resources and services by direct exchange between systems. These resources and services include the information, processing cycles, cache storage and disk storage. P2P computing takes advantage of existing computing power, computer storage and networking connectivity, allowing users to leverage their collective power to the 'benefit' of all. In P2P networks, nodes are autonomous and heterogeneous. This means that peers are not under control of a central authority and it can accommodate peers of different types, i.e. different operating systems, network connections, etc. Nodes collaborate directly with each other (not through well-known servers) and most traffic passes between peers. Nevertheless, peers mainly interact with each other and this accounts for most of the system traffic. P2P systems are overlay networks, because they run on top of another network (the Internet, for example).

P2P networks can be broadly classified into two types: Unstructured P2P networks and structured P2P networks [2]. The differentiation of structured and unstructured P2P networks is based on the topology of P2P overlay networks, ie whether the construction of overlay network is based on some special rules or it is constructed randomly. Unstructured networks organizes peers in a random graph, in a flat or hierarchical way and uses flooding or random walks for search on the graph. For locating an object, the peer broadcasts the request to its neighbors which again forward the request to their own neighbors. Here a peer can be a neighbor of any other peer in the network. eg: Gnuttella [3], Kazaa[4]. In structured P2P networks, topology is tightly constrained and content is placed at specified locations, rather than at random nodes and thus makes queries more efficient. Searching in structured P2P is deterministic and more scalable. Thus in structured P2P, peers form a rigid topology, which allows efficient mechanisms for locating objects. A number of structured overlays exist. eg: CAN[5], Chord[6] and Pastry [7].

II. P2P System Topologies

This section compares the organization of various P2P software system topologies [8] with respect to flow of control and data.

Pure P2P Networks

In a pure P2P network, all participating peers are equal, and each peer plays both the role of client and of server. Figure 1 depicts a pure P2P network where all nodes are considered to be equal. The system does not rely on a central server to help control, coordinate, or manage the exchanges among the peers. A primary virtue of pure P2P systems is their scalability; any node can join a network and start exchanging data with any other node. These systems also tend to be fault tolerant, as the failure or shutdown of any particular node does not impact the rest of the system. Gnutella [3] is an example of a pure P2P network.

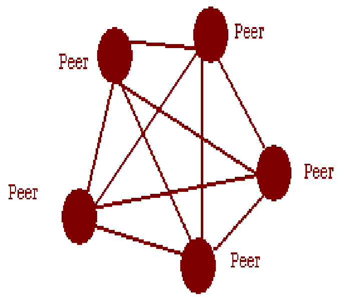

Figure 1: Pure P2P Network

Hybrid P2P Networks

In hybrid P2P systems the control information is exchanged through a central server, while data flow takes place in a pure P2P manner, ie a central server exists to perform certain "administrative" functions to facilitate P2P services This architecture alleviates the manageability problems of pure P2P systems. The control server acts as a monitoring agent for all other peers and ensures information coherence. The drawbacks associated with control being centrally managed still remain. If the central server goes down, the system looses ability to affect changes in data flow. However existing applications are not affected by a failure of the central server as the dataflow between nodes continues regardless of whether the central server is functional or not. Peer to peer data routing allows the hybrid system to offer better scalability than a centralized system; but hybrid systems still suffer from scalability problems for control information that flows through a single node. Figure 2 shows a hybrid P2P network. An example of hybrid P2P networks is BitTorrent [9] where a central server called a tracker helps coordinate communication among peers in order to complete a download

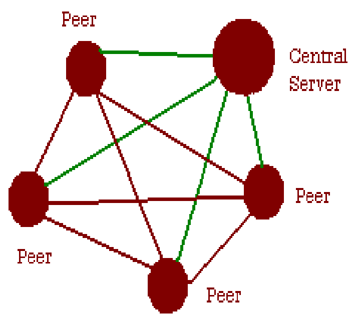

Figure 2: Hybrid P2P Network

Super Peer Architecture

A new wave of peer to peer systems is advancing architecture of centralized topology embedded in decentralized systems; such topology forms a super peer network. Figure 3 shows a super peer architecture for P2P networks. The important features of super peer architecture are:

- ***Reduced time and bandwidth for search** :*The search is much faster in super peer networks when compared to other topologies, since the system is now broken into a search of information from a smaller set of super peers, each of which have indexed information for their set of peers. For instance, a search which takes O(N) time on a pure/hybrid P2P network, will take O(N/M) time on a super peer network (where M is the average number of peers connected to a single super peer). This nearly eliminates the problem of network flooding typically associated with a pure P2P system.
- ***Manageability** :*Super peers, which are more reliable and trustworthy peers, can monitor client activity of all peers connected to them. This ensures that malicious activities can be controlled across the network.
- ***Load balancing** :*In a pure P2P network, every peer is given equal responsibility irrespective of its computing/ network capabilities. This can quickly lead to deterioration of performance due to network fragmentation as less capable nodes are added. This problem is alleviated in super peer architecture, as only relatively powerful computers with large network bandwidth are promoted to the status of super peers. This ensures that the super peer network divides load according to the capability of the peers, leading to overall better performance.

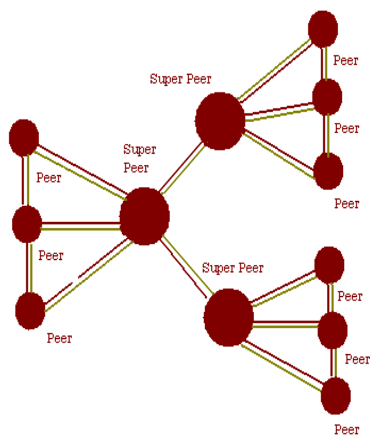

Figure 3: Super peer Architecture

Although super peer clusters are efficient, scalable and manageable, a super peer becomes a potential single point of failure for its clients. This problem is overcome via the notion of super peer redundancy, in which fail-over super peers are defined to automatically take over the job of the other super peer in case of failures. From the above discussions, it is clear that a redundant super peer architecture which combines the virtues of both centralized and decentralized systems, is the most suitable topology employed for developing and deploying distributed software systems.

III. SECURITY ISSUES CONCERNING P2P NETWORKS

When analyzing the various attacks that are possible on P2P systems, it is important to first understand the motivation of the attackers as well as the resources that they would have at their disposal. Once the threat has been identified, admission control [9] is a first step towards security that can help avoid a substantial number of attacks. Most solutions rely on the assumption that malicious nodes represent a small fraction of all peers. It is therefore important to restrict their number in the overlay. P2P systems lack the tools available to a centralized administrator, so it can be much more difficult to implement security protections on a deployed P2P system [10, 11, 12, 13]. Here P2P specific attacks are considered from two different planes: the data plane and the control plane. Attacking the data plane means attacking the data used by the P2P application itself, for example by poisoning it or rendering it in any way unavailable. On the other hand, attacking the control plane means directly attacking the functionality of the P2P application, trying to render it slower than or as inefficient as possible. Depending on the attacker's goal, he will choose to attack in one plane or the other, or both. These two planes are not completely independent. For instance by attacking on the data plane and corrupting many files, users will tend to download more instances of a file thus slowing down the traffic which is typically the aim of a control plane attack. Vice versa, eclipse attacks which are in the control plane can render data inaccessible, which is the primary objective of a data plane attack. The possibilities of attacks are enormous in P2P networks. Now follows a brief discussion on some of the common attacks in P2P networks.

Rational Attack

For P2P services to be effective, participating nodes must cooperate, but in most scenarios a node represents a self-interested party and cooperation can neither be expected nor enforced. A reasonable assumption is that a large fraction of P2P nodes are rational and will attempt to maximize their consumption of system resources while minimizing the use of their own [14]. For example nodes might realize that by not sharing, they save precious upload bandwidth. In the case of copyrighted material, file sharing can have worst outcomes. As it is illegal and quite easy for authorities to find out who is sharing specific files, it can lead to a very big fine. These are good enough reasons to motivate nodes in becoming "self-interested". If a large number of nodes are self-interested and refuse to contribute, the system may destabilize.

File Poisoning

File poisoning attacks [15, 16] operate on the data plane and have become extremely common in P2P networks. The goal of this attack is to replace a file in the network by a false one. This polluted file is of course of no use. In order to attack by file poisoning, malicious nodes will falsely claim owning a file, and upon a request will answer with a corrupt file. Moreover, all messages passing through malicious node can be poisoned. These factors may give the poisoned file a high availability, making it more attractive to download the true file. The reason that file-poisoning attacks are still successful today are due to the factors like clients are unwilling to share (rational attack), corrupted files are not removed from users machines fast enough and users give up downloading if the download seemingly stalls. These factors each give advantage in different ways to the most available file, which probably is the polluted file at the beginning.

Eclipse Attack

Before an attacker can launch an eclipse attack [17], he must gain control over a certain amount of nodes along strategic routing paths. Once he has achieved this, he can then separate the network in different subnetworks. Thus, if a node wants to communicate with a node from the other subnetwork, his message must at a certain point be routed through one of the attacker's nodes. The attacker thus "eclipses" each subnetwork from the other. In a way, eclipse attacks are high-scale man-in-the-middle attacks. Through this attack an attacker can attack the control plane by inefficiently rerouting each message, or he can decide to drop all messages he receives, thus completely separating both subnetworks, or he can attack the data plane by injecting polluted files or requesting polluted files on behalf of a innocent nodes and hoping, these files are cached or copied along the way.

Sybil Attack

Sybil attacks are part of the control plane category. The idea behind this attack is that a single malicious identity can present multiple identities, and thus gain control over part of the network. These Sybil identities can easily "out vote" the honest users. The Sybil attack can be used to destroy the integrity of reputation systems and may cheat P2P computing systems like SETI@home which uses voting to verify correct answers, but may accept false solutions from a Sybil attacker. Google's page rank algorithm is also influenced by Sybil attack. Unfortunately, without a central trusted authority, it is not possible to convincingly stop Sybil attacks. Maybe, carefully configured reputation-based systems might be able to slow the attack down, but it will not do much more. The best solution we currently have to this problem is to moderate the rate at which node identifiers is given out. Possible solutions include charging money in return for certificates or requiring some form of external authentication. While it may be possible to use some form of cryptographic puzzles, these still allow attackers with large computational resources to get a disproportionate number of node identifiers. An open problem is assigning random node identifiers without needing a centralized authority. Our thesis mainly concentrates on Sybil attack in P2P networks and remaining part of this chapter discusses different solutions to limit this attack.

IV. SYBIL DEFENSES

Sybil attack was first described by John Douceur [18] and he showed that without a logically centralised authority, Sybil attacks are always possible except under extreme and unrealistic assumptions of resource parity and co-ordination among peers. He argued that distributed approaches to defend the Sybil attack cannot be completely secure, and proposes a central authority certifying all identities concurrently as the only effective solution. An entity has three potential sources of information about other entities: a trusted agency, itself, or other untrusted entities. In the absence of a trusted authority, either an entity accepts only identities that it has directly validated (by some means) or it also accepts identities vouched for by other identities it has already accepted. For direct validation, he showed that even when severely resource constrained, a faulty entity can counterfeit a constant number of multiple identities. Also each correct entity must simultaneously validate all the identities it is presented; otherwise, a faulty entity can counterfeit an unbounded number of identities. Large-scale distributed systems are inevitably heterogeneous, leading to resource disparities that exacerbate the former result. The latter result presents a direct impediment to scalability. The only direct means by which two entities can convince a third entity that they are distinct is by performing some task that a single entity could not. If the assumption is that the resources of any two entities differ by at most a constant factor, a local entity can demand proof of a remote entity's resources before accepting its identity.

Thus, in addition to accepting identities that it has directly validated using one of the challenge mechanisms above, an entity might also accept identities that have been validated by a sufficient count of other identities that it has already accepted. If an entity that has presented identity *i1* claims to have accepted another entity's identity *i2*, we say that *i1* vouches for *i2*. An obvious danger of accepting indirectly validated identities is that a group of faulty entities can vouch for counterfeit identities. For indirect validation, in which an entity accepts identities that are vouched for by already accepted identities, he showed that a sufficiently large set of faulty entities can counterfeit an unbounded number of identities. Also all entities in the system must perform their identity validations concurrently; otherwise, a faulty entity can counterfeit a constant number of multiple identities. Since the number of faulty entities in the system is likely to grow as the system size increases, the former result places another limit on system scale. The latter restriction becomes harder to satisfy as system size increases. There is no general solution to Sybil attack. Many schemes have been proposed to limit Sybil attack over using social networks [19] to mitigate multiple identity, or Sybil attacks. These insights were initially made and leveraged by SybilGuard [20] and SybilLimit [21], which first introduced this approach and here these two methods are described in brief. Later these insights became the common foundation for many other sybil defenses via social networks like SybilInfer [22], Gatekeeper [23], SumUp [24] etc.

Sybil Defenses Leveraging Social Networks

Consider a distributed system *D* for which sybil defense is needed. The system has *n* honest human beings as honest users, each with one honest identity (e.g., one account). Honest identities obey the protocol. The system also has one or more malicious human beings as malicious users, each with one or more identities. To unify terminology, all identities created by malicious users are called as Sybil identities. Sybil identities are byzantine, and thus may behave and collude in adversarial ways. The goal of Sybil defense is to allow any given honest identity *V* to label any other given identity *S* as either "honest" or "Sybil" [25].

The social network for a distributed system *D* is an undirected graph *G* where the vertices of the graph are the identities in *D.* An edge between two nodes in *G* corresponds to human-established trust relations between the two corresponding identities in the real world. For example, it may have edges between two colleagues or two relatives. The honest region of *G* is defined to be *G's* subgraph consisting of all the honest nodes and edges among them. The edges in *G* connecting the honest region and the Sybil region are called the attack edges. (as shown in Figure 4).

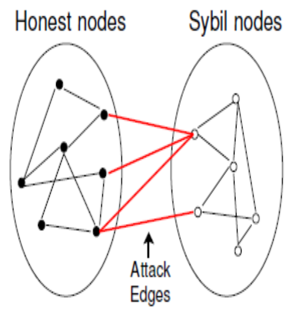

Figure 4: The Social network and Attack Edges

Information about the social network *G* might not be directly available in *D*, in which case the users need to provide enough information about *G* to the defense mechanism. If the defense mechanism is decentralized (such as Sybil Guard and Sybil Limit), then each node needs to tell its local instance of the defense protocol who are its neighbors in *G*. To do so, each pair of neighboring nodes in G needs to first establish a shared secret symmetric key, via out-of-band means. This key is called the edge key since it corresponds to an edge in *G*. Each node then feed all its edge keys to its local instance of the defense protocol. The purpose of these edge keys is to allow mutual authentication between a pair of neighbors when they communicate. Otherwise it is possible for one node *A* to communicate with another node *B* while faking as *B*'s friend *C*. If the defense mechanism is centralized, then the centralized server can directly collect information from each user regarding who are the user's neighbors in *G*. Care still must be taken (e.g., via out-of-band means) to ensure proper authentication. Otherwise, a malicious user may create an account *A* under the name of *B*'s friend *C*, and then fool *B* into forming an edge with *A*. This can be prevented if *B* authenticates *A* via out-of-band means.
Sybil defenses via social networks leverage the following three key insights on *G* to defend against Sybil attacks:

- The number of attack edges is independent of the number of Sybil identities, and is limited by the number of trust relation pairs between malicious users and honest users. This holds since each attack edge corresponds to a distinct edge key established between an honest node and a Sybil node controlled by a malicious user. Regardless how many Sybil identities that malicious user creates, it can only establish one edge key (via out-of-band means) with the honest user.
- If the malicious users create too many Sybil identities and given that the number of attack edges remains fixed, the cut along the attack edges will have a small quotient. Here the quotient is the quotient between the number of edges in the cut (i.e., the number of attack edges) and the number of nodes disconnected due to the cut. If we know beforehand that a social network with only honest nodes (e.g., the honest region in *G*) will not contain a cut with similarly small quotient, then we can tell that *G* is "abnormal".
- Finally, it is necessary to break symmetry in order to properly label nodes. For example, the two regions are completely symmetric; in which case we would still not know which region is honest after detecting the cut with small quotient. Breaking such symmetry is easy. If the local user running the protocol is honest, then the local protocol can safely claim that the region containing the local user is honest. If the local user is malicious, then do not care anyway.

Sybil guard protocol

Sybil Guard uses a special kind of random walks, called random routes, in the social network. In a standard random walk, at each hop, the current node selects a (uniformly) random edge to direct the walk. In random routes, each node uses a pre-computed random permutation as a one-to-one mapping from incoming edges to outgoing edges. Specifically, each node uses a randomized routing table to choose the next hop. For random routes in the honest region, the routing table gives the following properties. First, once two routes traverse the same edge along the same direction, they will merge and stay merged (called the convergence property). Furthermore, an outgoing edge uniquely determines an incoming edge as well; thus the random routes can be back-traced (called the back-traceable property). In other words, it is impossible for two routes to enter the same node along different edges but exit along the same direction. With these two properties, if we know that a random route *w* of a certain length traverses a certain edge *e* along a certain direction in its *i* th hop, the entire route *w* is uniquely determined. In other words, there can be only one route with length that traverses along the given direction at its *i* th hop. In addition, if two random routes ever share an edge in the same direction, then one of them must start in the middle of the other. Of course, these properties can be guaranteed only for the portions of a route that do not contain Sybil nodes. Sybil nodes may deviate from any aspect of the protocol.

In Sybil Guard, a node with degree *d* performs random *d* routes (starting from itself) of a certain length, one along each of its edges. These random routes form the basis of Sybil Guard whereby an honest node (the verifier *V*) decides whether or not to accept another node (the suspect *S*). In particular, a verifier route accepts *S* if and only if at least one route from *S* intersects that route from *V*. *S* accepts *V* if and only if at least a threshold *t* of *V*'s routes accept *S*. Because of the limited number of attack edges, if one chooses appropriately, most of the verifier's routes will remain entirely within the honest region with high probability.

To intersect with a verifier’s random route that remains entirely within the honest region, a Sybil node’s random route must traverse one of the attack edges (whether or not the Sybil nodes follow the protocol). Suppose there were only a single attack edge, based on the convergence property, the random routes from Sybil nodes must merge completely once they traverse the attack edge. Thus, all of these routes that intersect the verifier’s route will have the same intersection node; furthermore, they enter the intersection node along the same edge. The verifier thus considers all of these nodes to be in the same equivalence group, and hence there is only a single Sybil group. In the more general case of *g* attack edges, the number of Sybil groups is bounded by *g*. Thus an honest node’s random route intersects with the verifier’s route with high probability, and such an honest node will never compete for the same hop number with any other node (including Sybil nodes). Thus, the average honest node will be accepted with high probability.

Sybil Limit Protocol

Sybil Limit adopts a similar system model and attack model as Sybil Guard. The system has honest human beings as honest users, each with one honest identity/node. Honest nodes obey the protocol. The system also has one or more malicious human beings as malicious users, each with one or more identities/nodes. To unify terminology, we call all identities created by malicious users as Sybil identities/nodes. Sybil nodes are byzantine and may behave arbitrarily. All Sybil nodes are colluding and are controlled by an adversary. Compromised honest node is completely controlled by the adversary and hence is considered as a Sybil node and not as an honest node.

Consider a random walk in G, starting from some given honest node V and with some given length. The random walk is escaping if it ever crosses any attack edge. Otherwise it is non-escaping (shown in Fig 5).

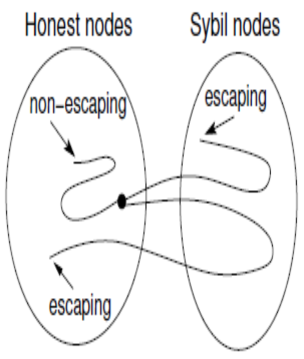

Figure 5 Escaping and Non-Escaping Random walks

Escaping random walks are bad “growth” since they get into the Sybil region. Even if an escaping random walk later comes back to the honest region, it is still bad because the Sybil nodes may have manipulated this random walk. Since they are byzantine, Sybil nodes may behave arbitrarily, and may not follow the random walk protocol at all. On the other hand, a non-escaping random walk is entirely in the honest region, and thus has nice probabilistic properties regardless of the byzantine behavior of the Sybil nodes. The escaping probability of a random walk starting from *V* is independent of the behavior of the Sybil nodes. But it does partly depend on the location of *V* in the honest region. If *V* is close to the attack edges, then the escaping probability will likely be higher. Social network based Sybil defense put forth a complex algorithm in detecting Sybil nodes. The main issue is obtaining the corresponding graph *G* where each node needs to tell its local instance of the defense protocol who is its neighbors in *G*.

Sybil Defenses Not Leveraging Social Networks

Many papers either suggest certification as a solution to the Sybil attack, following Douceur’s approach, or simply state the problem without giving a solution.

Trusted Certification

Douceur has proven that trusted certification [26] is the only approach that has the potential to completely eliminate Sybil attacks. Accordingly, it is cited as the most common solution. However, trusted certification relies on a centralized authority that must ensure that each entity is assigned exactly one identity, as indicated by possession of a certificate. In fact, Douceur offers no method of ensuring such uniqueness, and in practice it must be performed by a manual or in-person process. This may be costly and may create a performance bottleneck in large-scale systems. Moreover, to be effective, the certifying authority must ensure that lost or stolen identities are discovered and revoked. If the performance and security implications can be solved, then this approach can eliminate the Sybil attack.

Resource Testing

The goal of resource testing is an attempt to determine if a number of identities possess fewer resources than would be expected if they were independent. These tests include checks for computing ability, storage ability, and network bandwidth. In resource testing, identities are periodically revalidated using resource tests. This approach limits the number of Sybil attackers with constrained resources which may introduce attack in a period of time. However, in many applications few Sybil identities are required for an effective attack. Also computational power can be tested. Computational power mostly involves a one-time cost (for example, the purchase of computing hardware), so an attacker who could afford high initial costs can claim a large number of identities. Another approach is to use Turing tests, for example CAPTCHAs, to impose recurring fees. Also an identity may be obtained through computational cost, for instance with crypto-puzzles [27]. The problem is that an attacker can compute a huge number of identities before joining the network. One solution is that challenges are constantly refreshed, in order to avoid precomputation. Another method uses a tree to represent members and to distribute computational challenges from the root of the tree [36]. However, an attacker with enough computation power is still able to generate a huge amount of identities. The Douceur has proven the ineffectiveness of resource tests, but a number of researchers suggest them as a minimal Sybil attack defense. In these cases the stated goal is to discourage rather than prevent Sybil attacks, and the number of identities an attacker can have is, in theory, limited. For many applications this is insufficient if an attacker can obtain enough identities for a successful attack, even if it is expensive.

Recurring Costs and Fees

An identity may be related to some material cost, usually some amount of money. Identities can be linked to smartcards which are issued by a trusted third party or a fee can be charged to mitigate Sybil attacks. The problem in imposing fees [28] is the disparity between users and so the setup of an entrance barrier. This entrance barrier must be low enough to allow everyone to join, but must also be high enough to prevent some misbehaving user from buying many identities. Also IP addresses are used as a part of the identifier. Again, a huge number of IP addresses can be owned through paying. For many applications, recurring fees can incur a cost to the Sybil attacker.

Some Other Sybil Defenses

There are also a number of interesting sybil defenses [30,31,32,33,34,35] proposed recently which do not leverage social networks. A general sybil defense, was proposed by Bazzi and Konjevod [29] in which it is based on network coordinates. Their approach offers provable guarantees under certain assumption on the network position of the attacker. DSybil [36] uses user feedbacks to defend against sybil attacks in recommendation systems. DCast [37] uses pairwise entry fees to defend against sybil attacks in overlay multicast with rational players. Danezis et al. [38] propose exploiting the bootstrap tree of the DHT for defending against sybil attacks in DHT routing

Another approach is a referral system based on multiplicative reputation chains [39] in which it shows how a reputation system with chain referrals adds referrals from different referral paths/chains, is sybil-proof. In [40] an existing member has to invite another user for obtaining an identity in the network. This method is based on the construction of a perfect tree representing social relationship between users. The top of the tree consists of founding members which emulate root node by sharing private key of the root through threshold cryptography. The invitations are delivered based on the value of a factor parameter which is calculated by each node based on their local policies. Here the term 'weight' of a node corresponds to number of invited nodes and 'potential' corresponds to maximum attainable weight. The algorithm tries to construct a perfect tree in which, for each child, weight is equal to the potential and for each parent their children have a potential corresponding to the factor parameter. The performance depends on proper selection of value of the factor parameter.

V. CONCLUSION

P2P computing opens door to many new security threats especially in identity management. Distributed identity management is much more profound and demanding than that performed by a centralized server. Here information is more spatially distributed and ensuring its consistency is much harder. In addition, the possibility of unauthorized data access is much higher. In many applications user authentication is carried out through a designated centralized server while normal service is performed in a P2P fashion. In addition, identity authentication goals often conflict with privacy goals and trade-off are usually warranted. Identity assignment is also much more complicated in a P2P distributed system. Managing multiple versions of users' identities across multiple legacy applications makes the task more tedious. Thus Sybil attack is difficult to stop, but it can be limited to an extent using different techniques. This paper summarizes various approaches to Sybil attack in P2P networks.

REFERENCES

[1] Peer-to-Peer working group: What is peer-to-peer? http://www.peer-to-eerwg.org/whatis/index.html.
[2] Rüdiger Schollmeier, A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE 2002.
[3] The Gnuttella protocol specification v4.0. https://dss.clip2.com, Nov.6,2000.
[4] KaZaA Homepage, http://www.kazaa.com
[5] Ratnasamy et al. A Scalable Content-Addressable Network. In Proceedings of ACM SIGCOMM 2001.
[6] Stoica, Ion et al. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications" Proceedings of SIGCOMM'01 (ACM Press New York, NY, USA).

[7] A.Rowstron and P. Druschel (Nov 2001). "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany: 329–350.
[8] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Lab, 2002.
[9] Cohen, B. (2003) "Incentives Build Robustness in BitTorrent," in Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA.
[10] Yongdae Kim, Daniele Mazzochi, and Gene Tsudik. Admission control in peer groups. In Proceedings of 2nd IEEE International Symposium on Network Computing and Applications (NCA), pages 131–139. IEEE Computer Society, 2003.
[11] Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S.Wallach. Secure routing for structured peer-to-peer overlay networks. In Proc. OSDI, pages 299–314, Dec 2002
[12] D. S. Wallach. A survey of peer-to-peer security issues. In Proceedings of the International Symposium on Software Security (ISSS), Lecture Notes in Computer Science. Springer-Verlag, 2002.
[13] RFC 4981 - Survey of Research towards Robust Peer-to-Peer Networks.
[14] R. Gatti, S. Lewis, A. Ozment, T. Rayna, , and A. Serjantov. Sufficiently secure peer-to-peer networks. In Workshop on the Economics of Information Security, May 2004
[15] N. Christin, A. Weigend, and J. Chuang: Content availability, pollution and poisoning in peer-to-peer file sharing networks. In ACM E-Commerce Conference, 2005.
[16] J. Liang, R. Kumar, Y. Xi, and K. Ross : Pollution in p2p file sharing systems In IEEE INFOCOM, 2005.
[17] Atul Singh, Miguel Castro, Peter Druschel, Antony Rowstron: Defending against Eclipse attacks on overlay networks
[18] J. R. Douceur. The sybil attack. In Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS),volume 2429 of Lecture Notes in Computer Science, pages 251–260. Springer-Verlag, 2002.
[19] B. Viswanath, A. Post, K. Gummadi, and A. Mislove. An Analysis of Social Network-based Sybil Defenses. In SIGCOMM, August 2010.
[20] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In Proceedings of the ACM SIGCOMM Conference (SIGCOMM).ACM Press, 2006.
[21] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. Technical Report TRA2/08, National University of Singapore, School of Computing, March 2008.
[22] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil Nodes using Social Networks. In NDSS, 2009.
[23] N. Tran, J. Li, L. Subramanian, and S. Chow. Optimal Sybil-resilient Node Admission Control. In INFOCOM, Apr. 2011.
[24] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In NSDI, 2009.
[25] Haifeng Yu, Sybil Defenses via Social Networks: A Tutorial and Survey.
[26] Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S.Wallach. Secure routing for structured peer-to-peer overlay networks. In Proc. OSDI, pages 299–314, Dec 2002
[27] N. Borisov. Computational puzzles as sybil defenses. In Proceedings of the 6th IEEE International Conference on Peer-to-Peer Computing (P2P), volume 0, pages 171–176. IEEE Computer Society, 2006.
[28] H. Rowaihy, W. Enck, P. McDaniel, and T. L. Porta. Limiting sybil attacks in structured P2P networks. In INFOCOM, pages 2596–2600. IEEE, 2007.
[29] B. N. Levine, C. Shields, and N. B. Margolin. A survey of solutions to the sybil attack. Tech report 2006-052, University of Massachusetts Amherst, Amherst, MA, Oct 2006
[30] Dimiani, D., D.S. Capitani di Vimercati, and P. Samrati, Managing Multiple and Dependable Identities, In the proceeding of the IEEE Internet Computing, Published by the IEEE Computer Society, November/December 2003.
[31] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security," in Proc. Eurocrypt 2003, Warsaw, Poland, May 2003.
[32] Yu, B., M. Singh, and K. Sycara, "Developing trust in large-scale peer- to-peer systems," in Proc. of First IEEE Symposium on Multi- agent Security and Survivability, 2004.
[33] Selpk A.A, Uzun E, Pariente A, "Reputation-Based Trust Management System for P2P Networks", IEEE International Symposium on Cluster Computing and the Grid, 2004.FDGD
[34] Cheng and E. Friedman. Sybilproof reputation mechanisms. In ACM P2PEcon, 2005.
[35] F´elix G´omez, M´armol and Gregorio Mart´ınez P´erez. State of the Art in Trust and Reputation Models in P2P networks.
[36] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. DSybil: Optimal Sybil-Resistance for Recommendation Systems. In IEEE Symposium on Security and Privacy, May 2009.
[37] R. Bazzi and G. Konjevod. On the establishment of distinct identities in overlay networks. In ACM PODC, 2005.
[38] H. Yu, P. B. Gibbons, and C. Shi. Brief Announcement: Sustaining Collaboration in Multicast despite Rational Collusion. In PODC, 2011.
[39] G. Kesidis, A. Tangpong, C. Griffin, "A Sybil-proof Referal System Based on Multiplicative Reputation Chains, IEEE Communication Letters, 2009
[40] Lesueur, F., L. M´e and V. T. Tong. 2008. A Sybilproof Distributed Identity Management for P2P Networks, In Proceedings of the IEEE Symposium on Computers and Communications (ISCC), IEEE Computer Society, Morocco.