# 31. Security-Related Issues in Peer-to-Peer Networks

Luka Divac-Krnic, Ralf Ackermann
(Darmstadt University of Technology)

## 31.1 Introduction

The main characteristic of great autonomy of peers in Peer-to-Peer networks and the resulting "openness" of such networks makes them vulnerable to diverse attacks on their integrity and security. The possibility and the feasibility of obstruction of a Peer-to-Peer network as a whole, or forthright attacks on a single peer depend largely on a usage scenario of a Peer-to-Peer network. This aspect conditions the possibilities of attacks one has to either take care of or ignore.

When deciding what kind of threats to treat, one has to consider the tradeoff between the performance one wishes a particular Peer-to-Peer network to reach and the level of security one wants to achieve. Due to the different topologies Peer-to-Peer networks have, they face different kinds of threats which one has to consider.

The security concerns can be divided between at least two layers: breaches of security on the application layer and those on the networking layer. We will discuss security issues in the following two subchapters based on this differentiation.

## 31.2 Security Concerns on the Application Layer

In the following we specify the breaches of security that may take place on the application layer of a Peer-to-Peer network. This presupposes a direct act by a (malicious) user upon a Peer-to-Peer network through the application interface which enables direct user-to-network interaction. This can happen without a substantial effort by the user. Furthermore, we introduce the notion of a malicious node", meaning a node in the Peer-to-Peer network (and a user behind it) that uses the network improperly, whether deliberately or not.

Malicious nodes on the application layer may give incorrect responses to requests. They might report falsely about their bandwidth capacity in order, e.g., to not have much traffic routed over their own node or they might (in the case of file sharing) freeride in the network [11].

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 529-545, 2005.

These problems are usually met with incentives, where peers are rewarded for good behavior and penalized for bad. Virtual money for file exchanges was introduced by the now defunct Mojo Nation project [425]. Credits are given to peers in eMule [589] as a reward for files being uploaded and used as an entitlement for raising one's priority in the upload queue of another peer-client.

Following are security issues that arise from different applications of Peer-to-Peer networks.

### 31.2.1 File Sharing Applications

In a completely decentralized and open Peer-to-Peer network, it is relatively easy to disseminate spurious files that do not contain what their names or metadata suggest, but instead random content or binary data. Thus users get annoyed by contents that they were not looking for, and wasted their resources to download them or cache them temporarily. These files can be arbitrary files tagged with names that imply content that is usually searched for in a particular network.

As stated in [463] some individuals are hired and paid by IFPI (International Federation of the Phonographic Industry) [316] to fill the file-sharing networks with files with legitimate titles, but containing silence or random noise. This process is called "network poisoning" and such files are referred to as "junk" or "rogue" files. The aim was to make networks so disreputable, that casual users would give up downloading files.

The reason for these actions is of course the copyright issues. Guidelines for what kinds of files may or may not be placed in the network can be easily set for a small group of users, e.g., inside of a company, but it is nearly impossible to enforce them in an open network that spans multiple domains.

Furthermore, those downloading files from unknown peers exposes themselves to infection with viruses. "Mandragora" was a virus made by Spanish virus-developers back in 2000 [371]. It was merely 8 kB and was exploiting the possibility of blanking the file extensions in Windows computers. The virus would scan the searches of a Gnutella client, intercepting one and reporting itself as a successful hit. When downloaded by another peer not noticing a very small file and not realizing that it was executable, it just installed itself on the new computer. This virus was not meant to do any damage, but only to expose the weaknesses of the Gnutella file sharing network.

### 31.2.2 Data Backup Service

One of the primary tasks in any kind of company environment is to efficiently and reliably ensure data management and backup through automated peri-

odic copies. The traditional backup approach in companies is to use a backup server to maintain backup copies of data across the whole company. Consequently backup servers are high-end, powerful, and expensive data storage devices. Considering the large expenses for administration of these and the usually low hard-disk utilization of individual workstations, a Peer-to-Peer-based backup solution would save some of the backup budget of a company. Several designs like [63], [386] or [134] deal with this topic.

There are basically two kinds of security issues here. One is the existence of sensitive data of certain peers that should not be visible to other peers in a network. This is a unique characteristic of a Peer-to-Peer backup system; a client-server solution with a backup server would not have to deal with such issues. A possible solution is to encrypt all the backups with a secret key which only the owner-peer knows, but in the event of crash the passwords written on it would also be lost. Taking into account the additional complexity for users of such system, the sensitive data should instead be backed up individually by users themselves using some traditional techniques like burning data on digital media carriers such as CDs.

The other security issue affects the performance of a Peer-to-Peer backup solution. Considering the variable availability of single workstations, a backup would have to be saved on more than one workstation to ensure its availability with a high probability in case of a crash. If there are, e.g., four backup copies of each file, the peer-workstations taking part in Peer-to-Peer backup would have to have a utilization of their hard disks of approximately 20 percent in order to have the remaining 80 percent used by the backup application. Disregarding this constraint, the system might not have enough resources for smooth functioning and data might be lost.

This possibility can be regarded as a "violation of an agreement" between peers rather than an intentional attack on the system's resources. If hard disk capacities continue to grow, then this specific problem will be mitigated. However, this solution is more applicable for backup of text-based files than for large multimedia content, because of the unique restriction of Peer-to-Peer backup described above.

### 31.2.3 File Storage Service

The file storage service provides the storage of files on more than one location in a Peer-to-Peer network by using a uniquely defined e.g. alphanumeric character sequence for each file, called "handle". Contrary to file-sharing mechanisms where users are looking for specific content, the searches in the file storage system can be accomplished only with help of these specific handles which are mapped to a unique peer or a file in the network. Any peer in the network can locate a file by using it, and the peer with writing rights can update these files as well. In order to get a handle to find a file, a peer has

to obtain it through some other communication path than the Peer-to-Peer network itself, like an e-mail exchange, an external website, or even through out-of-band communication.

A file storage application may be used to enable an anonymous and decentralized publication of contents (chapter 31.4.4). The security concern here is the enforcement of an access control among peers in regard to read/write rights on files. An access control list may be associated with each handle provided that all peers behave well. Once malicious peers participate in the network, the files would have to be encrypted with secret keys (chapter 31.4.3) and provided only to a smaller set of peers that are proven legitimate. In anonymizing solutions (chapter 31.4.4) attacks could be directed at making the files unavailable to legitimate peers by deleting them altogether.

## 31.3 Security Concerns on the Networking Layer

As Peer-to-Peer systems distribute resources and responsibilities into the network, they also distribute security weaknesses allowing malicious peers to take advantage of the vulnerabilities of the system. Malicious peers in a Peer-to-Peer system are those which do not follow correctly the protocol which peers use to administer themselves in a Peer-to-Peer network. They seek to misguide other peers by providing them with false information about the state of the network.

These attacks are mostly relevant in structured Peer-to-Peer networks that use distributed hash table (DHT) lookup. The reason is their structure's vulnerability to false information about the routing in the network: the nodes are mutually dependent on the correctness of each other's routing tables.

### 31.3.1 Invalid Lookup

A malicious node may forward lookups to an invalid node, a non-existing node or an existing but random node. Incorrect lookups would result in a waste of bandwidth, time, and efficiency of a lookup until the TTL (time-to-live) runs to zero and the requestor ceases to retransmit its queries. Deriving node identifiers by a cryptographic hash of its IP address and a port, like in Chord [576], can make it easy to tell if the correct node is approached since these are needed to contact the node. By deriving node identities from public keys of participants, a longer-term solution may be reached [565]. This solution has performance disadvantages due to the cost of signatures, but provides validity of the origin of a message.

### 31.3.2 Invalid Routing Update

Nodes in a distributed hash table (DHT) lookup system build their routing tables by consulting other nodes. A malicious node here may corrupt the routing tables of others by sending invalid updates. In this case not only could malicious nodes direct queries to invalid nodes, but well-behaving nodes might give wrong routing information as well, due to invalid routing updates of their own. A more sophisticated attack would be to provide nodes that actually contradict general criteria of routing (high latency, low bandwidth). CAN [504] uses measurements of RTT (round-trip-time) to favor lower latency paths in routing updates. This method may be used for choosing nodes with high latency path as well.

Pastry [527] makes these invalid routing tables easier to detect by forcing entries to have a correct prefix. This way malicious nodes can not produce totally random routing tables without being easily detected. Another way of ensuring that a given node in an update table is authentic, is for every node to ascertain that this node is actually reachable [565]. This method would however cause enormous costs in bandwidth and delay.

### 31.3.3 Partition

When a node contacts another node in order to join a Peer-to-Peer network, it might falsely join a parallel malicious network which aims to observe the particular behavior of some nodes or contents of a network by running the same protocols as the legitimate one [565]. By providing even the look up capability into the original network and having a node connecting both of these networks or having some original data from the legitimate one, this coalition of malicious nodes could manage to "camouflage" itself and its true intent. A new node might feel itself well off in the network, whereas it is actually subject to observation of its actions which are supposed to be kept private.

The use of public keys, though expensive, can establish some stronger identity than mere IP-addresses, which are subject to frequent changes. If a node has successfully joined a legitimate network in the past, it possesses certain knowledge about the network and possibly nodes that successfully answered its queries. It can indirectly check the routing tables of other nodes, by performing random queries to gain knowledge about possible different views of the network than its own.

### 31.3.4 Sybil Attack

The Sybil Attack is a name for a node or any other single entity on a network presenting multiple identities to other nodes. Thus other nodes might believe in having an interaction with some distinct nodes whereas in fact there is only one entity they are addressing. This would mean that given particularly huge resources (bandwidth, disk space, computing power) some "deceiving" peer could gain control of a large part of a Peer-to-Peer network, thus undermining its redundancy [177], which is one of its basic properties. However, considering a Peer-to-Peer network to be made up of nodes at the "edges of a network" [28], one could assume rather moderate resources at the disposition of each peer. In a network of several hundred thousand nodes, such a peer could presumably cause rather little damage.

In order to prevent the generation of multiple identities in a Peer-to-Peer network, computational puzzles, like in HashCash [41], might be used. This is an old solution for defense from DoS (denial-of-service) attacks. Before joining the network a peer has to solve some computational problem, thus is forced to use his CPU-cycles needing more time to join than usual. But in the case of an attacker with huge resources, it would at best slow down the process of generation of false identities and the process of these "virtual" nodes joining the network. In structured Peer-to-Peer networks like Chord, CFS, and Pastry where the network determines different tasks that certain peers have to do, hashing of IP addresses is performed partly to establish some kind of identity of individual peers. This would surely complicate and prolong the process of a Sybil Attack.

The most "intrusive" method of binding an identity directly to a node would be, of course, to provide a distinctive identification for each computing unit that is taking part in a network. The commercial platform EMBASSY [292] provides cryptographic keys embedded inside of every hardware device. Of course a user has to implicitly trust that these devices have an embedded key (and not an arbitrary one) and the users actually use these as they are supposed to. The concept of Pretty Good Privacy (PGP) [644] web-of-trust may vouch for established identities for other "newcomers" in the network, but may also be misused by the malicious node to subvert the chain of trust. As a last resort for keeping all the possible weaknesses of the aforementioned solutions somewhat under control, one can rely on certification authorities and similar trusted agencies.

As stated in [171] "one can have, some claim, as many electronic personas as one has time and energy to create". Primarily this energy can be a decisive constraining factor for success in such an attempt.

### 31.3.5 Consideration of Implications of Topology

Structured networks like nodes in a Chord [575] ring impose an additional burden on the available bandwidth of nodes. A considerable communication overload occurs, which makes nodes vulnerable in ways they would not be in a fully decentralized network like Gnutella [250].

Each instance of join and leave a structured network causes a DHT to rebalance its keys and the data among the nodes which generates a considerable traffic load and degrades the performance of the system [565]. In a Gnutella network this kind of overload is feasible with a sudden increase of search messages which will flood the network, as its search-protocol does not scale, so this condition would be considered standard. In DHT systems this excess data transfer and control traffic would render the whole network inefficient and impair the secure functioning of the system.

In a classic denial-of-service attack (DoS) where an attacker generates arbitrary packets and overloads a targeted node, it would seem as if the targeted node failed in a normal way. As every node in a DHT is responsible for certain data, some of it would be unavailable for a certain period of time. The possible damage here or in a decentralized Peer-to-Peer network will however seem much smaller than in centralized Peer-to-Peer networks like the former Napster or a hybrid Peer-to-Peer network with super-peers. By having the servers attacked with a DoS, whole portions of the network would be "knocked out" for a certain time.

In order not to become too suspicious to many other nodes at once and to make its own malicious behavior more difficult to detect, a node may behave well with some part of the peers in a network and follow the protocol correctly. In a DHT system those peers would be the immediate ones in the vicinity of the malicious node's identifier space [565]. Good behavior with these peers guarantees, at least, longer connectivity to the whole Peer-to-Peer network, as these peers would keep the malicious node in their routing tables. Legitimate reports of bad nodes would not be distinguishable from false accusations of the good ones. Public keys and digital signatures here would help gain some assurance in the legitimacy of the reports by having all peers sign their responses.

In conclusion, a structured topology of nodes provides considerable improvement in performance of the network as well as sensitivity to abrupt changes in the protocol behavior due to the possible impact of malicious nodes.

## 31.4 Security Concepts for Selected Systems

In following sections we will introduce some solutions which deal with security in the Peer-to-Peer area in quite different ways. The first two address security

in a "traditional" manner. This means ensuring that basic "building blocks" of security are met, such as having a protected privacy of communication and maintaining the integrity of messages. The other solutions aim at using the characteristics of a Peer-to-Peer network for ensuring the anonymity of users behind the network rather than a secure data flow among nodes.

The single most technologically advanced solution is "Groove", which is discussed in the next section. A system called "SixFour" is subsequently described, which is an example of a most rudimentary solution for ensuring confidentiality of an electronic data exchange. Finally, we will introduce "Freenet" and other anonymizing solutions.

### 31.4.1 Groove

"Groove", of Groove Networks Inc. [261], is a Peer-to-Peer groupware tool. It concentrates on providing extensive support for the collaboration of participants on a group-project. It offers so-called "workspaces" for aggregation of documents, messages, and application-specific data which are shared confidentially and unaltered among group-members. Groove especially serves the purpose of synchronizing documents and incremental parts of documents in a secure way.

It does not need any administration and acts like an ad hoc virtual VPN (virtual private network). Bootstrapping the whole constitution of a group is accomplished by one participant inviting others to join the group. It orients around already mentioned "workspaces", where all the messages and documents related to any collaboration activities are saved.

The documents in these workspaces or even parts of these documents, like incremental changes during some processing, are synchronized among all peers. One is able to work offline and upon reconnecting, the system will automatically synchronize the changes. This means that all changes will be transmitted with help of so-called "delta messages" to all Groove-peers taking part in a shared space.

Groove offers, besides the permanent secured connection, different applications on its platform as well, like permanent chat sessions, notepads, a calendar, and a file archive. Besides offering an infrastructure for communication, it is a commercial product oriented towards the needs of users collaborating on a project and offers different add-on components of its software such as Enterprise Management Servers, which can take some of the decentralized tasks and administer them centrally.

All shared space data is encrypted on the hard disks as well, where it is only readable and editable by its owner. This security configuration is always on: it cannot be switched off by an administrator or accidentally by a random peer. Groove provides authentication, data privacy, and data integrity.

Groove has chosen Component Object Model (COM) as its component integration model. COM is Microsoft's framework for developing and supporting program component objects. The Groove Development Kit (GDK) is free for further development. For the distribution or selling of software developed with the GDK one needs an additional licence from Groove Networks Inc.

Groove uses the following basic encryption technologies:

- Symmetric Key Encryption: the standard encryption algorithms are AES and DES.
- Public Key Technology: the standard encryption algorithm is RSA-1; used for securely sending symmetric encryption keys over the network and for signing documents.
- Hashing: the standard encryption algorithm is SHA-1; it is used for password management (password hashed into a secret key), integrity protection (storing a hash of data along with data), and signatures (hashing data to be signed).
- Message Authentication Code (MAC): the standard encryption algorithm is Keyed-Hash Message Authentication Code (HMAC); it is used for signatures for data integrity; a key is used in the MAC computation when communication participants share a secret key.

**Authentication**

Groove's security model is based on an authentication mechanism that binds a user's identity to specific actions within a Groove environment. Generally, Groove depends on relationships that are rooted in a real-world collaboration because the very forming of a working space begins with an invitation (via e-mail, a telephone call, or face-to-face contact) to a user to join a specific group of people, whose relationships are represented through membership in a workspace.

Before being invited to a workspace, a user must configure his Groove software on his local device by creating his own account, including attributes describing him as a user. This account is stored as an encrypted XML-object. An account may contain more than one identity, e.g., one for personal and one for professional use. Each of these identities will possess sets of keys for communication with users in different workspaces.

Groove offers several ways for members of a workspace to authenticate others members of the same workspace. One of them is fingerprinting of the users' public keys with SHA-1. This method includes an out-of-band contact between the members of a workspace to confirm that the fingerprint is correct. Groove can also use its own PKI where a so-called Management Server would act as a CA (certification authority) for its own domain.

Besides authentication of persons, Groove performs data authentication as well. This is accomplished by using public key signature technology to "mark" the communication between peers and thus map actions in the workspaces to digital identities. Instant messages are signed with 2048-bit RSA keys and incremental changes in workspaces are signed with 1536-bit ESIGN keys.

A login into a Groove network may proceed either by a password or by a smart card. Both these logins are protected locally on the hard-disk with different keys: User Key and Master Key. User's password is processed through a PBKDF2 algorithm (Password-Based Key Derivation Function 2, RFC 2828) that has additional parameters to hamper guessing attacks. There are two master keys: the first protects the user's other cryptographic keys (like workspace keys) and the second protects non-cryptographic user data (like tool and contact data).

A user is obtaining relevant data from two databases when participating in Groove. The account database contains personal contact information for each identity (name, phone number, e-mail, public parts of RSA or ElGammal keys) and private keys associated with the identities held by the account.

**Workspace Security**

The beginning of a collaboration of peers in a workspace is accomplished by one peer inviting another to join the workspace he just set up. As stated above Groove depends on real-world relationships, so an invitation will be directed at someone personally well known. An invitee who does not yet have Groove software may be invited by an e-mail which will contain a URL of where to download it, public keys of the inviting peer, and cryptographic protocol settings for the workspace the invitee shall join. After receiving this e-mail, the invitee should phone the peer inviting and authenticate each other by checking their fingerprints.

After confirmation by the inviting peer, all other members in the workspace will get to know about the invitee as a new member by a so-called "add member delta" message and the invitee will receive the workspace with shared keys used by other peers. From then on the invitee's identity will appear on each of the other user's workspaces and by trusting the inviting peer they might begin working with their new member. The "add member delta" message contains data about the new member (ESIGN public key, Diffie-Hellman public key, and other information) and makes the old members aware of a new member joining them.

There is always someone who has to actually make a first step and kick-start a project on which others will collaborate: Groove uses for this purpose a hierarchy of roles and access controls. The three basic roles are: manager, participant, and guest (with declining access rights). The initiator of a new workspace automatically gets the role of manager and can therefore assign roles to others and invite or even un-invite peers to or from the workspace.

The roles themselves map to certain permissions related to actions in the workspace and actions with tools in the workspace.

By uninviting a peer from a workspace, which only the manager of the workspace may do, a "rekey delta message" is sent to all peers in the workspace except to the uninvitee, which contains the new workspace group key encrypted in each member's pair-wise key. The uninvited peer gets a delta message informing him he is uninvited and removes all the data from his workspace.

### Communication over Relay Servers

With "relay services" users may exchange changes in workspace with a time gap between the two actions. When a user goes offline the relay service will "notice" it, save the changes, and pass them to a partner peer of the same workspace that may be offline when this peer gets online again.

Groove software randomly chooses a relay server from a list it contains and registers a new account that a user previously set up. From now on, the software uses only this relay server and sends, along with its contact information, also the URL of its relay server in order for other Groove peers to acquire the full communication path. More precisely, an actual relay-to-workspace communication is taking place with a secret key established between them, whereas due to end-to-end encryption the relay service can not access data inside delta or instant messages but only the header information needed to locate another peer.

Furthermore, relay servers can enclose the messages in an HTTP - compatible format and use ports 80 or 443 (mostly configured open) to circumvent the firewalls that may separate two peers collaborating over Groove. This capability may well arouse suspicion with managers of the company or network administrators of a company's LAN who may not approve of arbitrary data being exchanged across the company's boundaries.

### Conclusion

Without relay servers Groove's ad hoc synchronization of data made offline would not work. This centralized feature plays an important role in an effective collaboration in Groove, so the Peer-to-Peer paradigm of fully autonomous entities is constrained here. Such a server is still a single point of failure and a potential target for a DoS attack. Furthermore, at the deepest level of integration, any platform that wishes to integrate and further develop Groove will have to support the Component Object Model (COM).

### 31.4.2 SixFour (6/4) Peer-to-Peer

The SixFour Peer-to-Peer system [566] obtained its name from the date of the Tiananmen Square protests in Beijing in 1989. The developers intended this system for potential users who live in oppressive countries with limited access to free press. The basics of the system enable a peer to establish an encrypted connection to a trusted peer outside of his country and then have this peer further forward any other requests into the Internet. The aim is to enable peers to get data confidentially from the Internet thus evading censorship.

A peer wanting to connect to the SixFour network would already have to know the address of the trusted peer and his RSA public key. The authenticity of the key would have to be checked at the website of the SixFour developers, hacktivismo.com. For this purpose the peers would have to know the appropriate signature of Hacktivismo [268].

The trusted peer simply has to forward the requests of peers that "hang" on them so they can access any TCP- or UDP-based service on the Internet, provided the trusted peer itself has access to these services, too. One of the important criteria for becoming a trusted peer is to provide a permanent IP-connection (permanent IPv4 address). The most reasonable entities to become trusted peers (according to the idea of developers) seem to be, e.g., human rights organizations or NGOs promoting democratic values.

The routing inside of the SixFour network is anonymous if it occurs over more than 3 nodes. That is, every peer in the routing protocol knows only the RoutingID, the source, and the target of the packet. The routing topology of the SixFour network is like Gnutella's, in that every peer is connected with several others and floods his own requests or routes other requests at random to his neighbors. Duplicate requests are rejected according to their RoutingID. The node-to-node encryption is a classical SSL over port 443 and the end-to-end encryption is RSA-based.

The shortcomings of this system are primarily the question of the establishment of the relationship between a peer inside a "censored" part of the Internet and a trusted peer "outside". How does a potential peer inside a "censored" part of the Internet come to know which IP-address is an address of a trusted peer, through which he could make requests and not be afraid of possible reprimand from the "censor"? It seems that only some kind of out-of-band information gathering would help.

Furthermore, it is not clear how these trusted peers could be protected from possible attacks from entities that would have an interest in disrupting the SixFour network (like those very countries whose censorship SixFour tries to break). The possible attackers on trusted peers presumably have the same or even better know-how than the developers of any free software on the market and possibly strong financial backing as well. An additional criterion before accepting entities acting as trusted peers could be a display of security measures that these peers employ locally and set requirements for.

Otherwise one would never know which trusted peer could be compromised, thus compromising the whole network.

Before becoming a serious application with wide acceptance, SixFour would have to establish a wider community of users and proof of a longer stable functioning (without security flaws and compromise of peers). It offers the basic functionality of confidential data exchange and anonymity in the network, but it does not tackle at all the question of key distribution. Although at the beginning of a possible future enhancement, SixFour is a fresh example of what Peer-to-Peer-systems are good for beyond simple file-sharing.

### 31.4.3 Freenet

Freenet [124] is a Peer-to-Peer system that enables publication, retrieval and replication of data whose authors and readers remain anonymous: a node can not know whether the node to which a file is forwarded is the actual requestor or itself a mere "forwarder" of the file. From another point of view a node can not know whether the node, from which it is obtaining a file is the actual originator or merely a forwarder of the file. Trust and adequate search of the network are still open issues in this system.

The cornerstones of the architecture of this system are the knowledge of the nodes of the immediate neighbors (routing tables are not exchanged between nodes), association of files with hash-keys, and distributed storage of files (the owners of a hard drive do not know implicitly what is stored on their site). The emphasis of the system lies heavily on the protection of privacy of users; in this case the disconnection of an association of a file with its originator or submitter and a possible user (reader) of the same file. The primary purpose of the network as a whole is censorship-resistant storage.

The files are replicated in the parts of network where they are most frequently requested and deleted in those parts of the network where they are seldom requested. Freenet acts like a distributed file system with location independence and transparent replication.

#### Security Architecture

Every Freenet peer runs a node that provides some storage space for the network. When adding a file, a user assigns a globally unique identifier (GUID) to that file. This GUID will be sent when retrieving a file as well. The GUID is calculated using the SHA-1 hash and made up of 2 keys: the content-hash key (CHK) and the signed-subspace key (SSK). CHKs are hashed contents of the files using the SHA-1. SSK is used to make up a specific namespace that anyone (who has the keys) can read, but only its owner can write to. A peer first generates an arbitrary public-private key and chooses a text description

of the subspace. One then calculates the SSK by hashing both the public part of the key and the descriptive string, concatenating them, and hashing them again.

In order to retrieve the file, one needs the public key and the descriptive string (to recreate the SSK). To add to, or update the file, one would need the private part of the key to be able to compute the valid signature, which the nodes storing the file would check and accept, or reject if false. The SSK key can be used to store an indirect file containing a pointer to a CHK: a file is stored under its CHK, which is in turn stored as an indirect file under the SSK. Given the SSK the original file is retrieved in 2 steps, and the origin of the file is obscured a step further.

One of the most distinctive characteristics of Freenet is that management of the node and the management of the storage of the same node is somewhat disjointed. If a node gets a query, it first checks it own store. The peculiarity here is that the semantics of the content of the store itself is not comprehensible to humans. The comparison between the request and the possible file in the storage has to be computed.

If the request is not satisfied, the node forwards the request to the node with the closest key to the one requested. This information will be gathered at one specific node through time by having many requests running through it. When the request is successful, each node which passed the request now passes the file back and creates an entry in its routing table binding the data holder with the requested key. On the way back the nodes might cache the file at their stores. This way subsequent searches find the requested file faster.

For requesting and inserting a file, Freenet offers the possibility of adding a mix-style "pre-routing" of messages [124]. This way the messages would be encrypted by a succession of public keys which establish a route that this message will follow. When the message reaches the end-point of this pre-routing path it is injected into the Freenet network and thus the true originator of the message is "obliterated".

Inserting a file works similar to requests. A user assigns a GUID and sends an insert message first to his own node with a new key and a TTL value. The insert might fail because the same file is already in the network (CHK collision) or there is a different file with the same description (SSK collision). The checking for a same possible file refers to a lookup whether the key already exists; if not, the node searches for the numerically closest key and forwards the insert message to the corresponding node. If the TTL expires without collision the final node sends the message that the insert may be performed. On the way to the final node, the file is cached by every intermediary node, the data is verified against the GUID and a routing entry is made pointing to the final node as the data holder.

The makers of Freenet recommend encrypting all data before inserting them into the network. Since the network does not perceive this encryption since it only forwards already encrypted bits, the inserters have to distribute

the secret keys as well as the corresponding GUIDs directly to the end users. This is performed through some out-of-band means such as personal communication. The same method is used when adding a new node to Freenet: a user wishing to join sends his new public key and a physical address to a node whose physical address he already knows.

In order not to let only one node decide what key to assign to a joining node (and allow a sort of unilateral access to certain data) a chain of hashes of the seeds of each node and XOR'ed results with other seeds down the path and hashes thereof (called "commitments") is produced to provide the means for every node to check if other nodes revealed their seeds truthfully. The key for a joining node is assigned as the XOR of all the seeds.

A still open issue is search adequacy of the Freenet network for relevant keys. There is still no effective way to route searches, leaving the dissemination of keys solely to out-of-band means. One possible solution is to build public subspaces for indirect keyword files. When inserting files, one could insert several indirect files corresponding to search keywords for the original file, so an indirect file would have pointers to more than one file (more than one hashed key as content).

The management of storage determines how long a file will be kept by the popularity of the file, measured by the frequency of the requests per file. Files that are seldom requested are deleted when a new file has to be inserted. Even when a file is deleted at one node, another one may still have a copy of it. The node that already deleted it, will still have an entry in its routing table pointing to the original data holder, as the routing tables entries are much smaller than data and will be kept longer.

### 31.4.4 Further Peer-to-Peer Anonymizing Solutions

Following are several more solutions that have as a priority the anonymization of a connection, transaction, data stream, or communication between two peers in a network. These systems do not protect specific data from access by an unauthorized peer or intrusions into peer-machines. They try to make facets of communication between two peers invisible or not back-traceable by a supposed eavesdropper. Usually the high connectivity of nodes in a Peer-to-Peer network is used to obscure the path an arbitrary message has taken.

#### Tarzan

Tarzan [229] determines sequences of mix-style relays randomly chosen from a pool of volunteer peers in the network. The relays are all equal peers which can be originators or relays of traffic, since they can not know whether they are the first hop in a path. The two ends of a communication session inside the Tarzan

network are a node running an application and another node running a NAT that forwards the traffic to an end destination. Tarzan performs a nested encryption per hop and encapsulates it in a UDP packet. These encryptions are performed in the sequence of the nodes that will be passed through, so the biggest share of encryptions is situated at the node seeking anonymity. The method is similar to Onion routing [582] with the difference that the nodes are chosen randomly and dynamically instead of using a fixed set. In order to choose the relays, Tarzan uses the Chord lookup algorithm (relays are in a Chord ring) with a random lookup key. When that key's successor is found, it responds with its IP address and public key. Tarzan claims thus to provide anonymity in cases of malicious Tarzan participants, inquisitive servers on the Internet, and observers with some limited capability to see traffic on some links.

### Publius

Publius [611] claims to be a censorship-resistant web publishing system. The motivation for this system is the possible, presumably unfair legal attacks from powerful corporate entities or similar litigious organizations which try to prevent publishing of "unpleasant" details about them or practices related to them.

A system-wide list of available static servers is assumed. The content is encrypted by the publisher and spread over some of the web servers. The publisher takes the key, K, that is used to encrypt the file and splits it into n shares, such that any k of them can reproduce the original K, but k-1 give no hints how to reproduce the key. Each server receives the encrypted content and one of the shares. At this point, the server has no idea what it is hosting; it is simply a store of random-looking data. To browse content, a retriever must get the encrypted content from some server and k of the shares.

The publishing process produces a special URL that is used to recover the data and the shares and that is cryptographically tied to the URL so that any modification to the content or the URL results in the retriever being unable to find the information, or a failed verification. Only publishers can update or delete their Publius content. The Peer-to-Peer character of this system is limited because of static IP addresses and the static set of servers. However, the content is in fact distributed and the servers which host the content are autonomous.

### Onion Routing

Onion Routing [582] is a system for anonymous Internet connections based on mix-networks. A user creates a layered data structure called an "onion" that specifies the encryption algorithms and keys to be used as data is transported to the target node. As data passes through each router on the way to the

final target, one layer of encryption is removed according to a directive found on the outer layer of the "onion". So the onion gets "peeled off" a layer every time it passes through a router. The final target of the "onion" gets only plain text (data).

#### Crowds

Users in this system [508] submit their requests through a "crowd" - a group of Web surfers running the Crowds software. Crowds users forward HTTP requests to another randomly selected member of their crowd. A path is configured as the request crosses the network and each Crowds member encrypts the request for the next member on the path, so the path is not predetermined before it is submitted to the network. Thus an end server or the forwarding Crowds member cannot know where a request really originated. By using the symmetric ciphers Crowds claims to perform better than mix-based solutions. A drawback of Crowds is that each peer on the path of the data to the intended destination can "see" the plain text.

## 31.5 Conclusion

The large amount of security vulnerabilities in Peer-to-Peer networks surely needs a considerable monitoring by all peers that are interested in the unobstructed working of the network. As already stated, there is always a tradeoff in Peer-to-Peer networks between security and performance issues.

Peer-to-Peer networks are supposed to be open to every peer that wishes to voluntarily provide his resources and in return consume the resources of others. Authenticating every one of them and signing all their messages would create a new overload which could bring the performance of a network nearly to a standstill. In accordance with its open character a Peer-to-Peer networks' primary concerns are not "classic" security measures, but the smooth functioning of a protocol that binds the peers in a network. This is what security concerns in Peer-to-Peer networks should focus on.

Though Groove orients itself toward one specific use of its network, it seems to be an up-to-date solution for classic security management. The use of relay servers diminishes somewhat a "real" Peer-to-Peer character, but the standard security features work just as well without them.

The anonymization tools constitute an "instrument" that the abundance of autonomous nodes with an adequate innovative communication protocol may establish. However they may always be a focus of social debate whether providing such means to keep the identity of wrongdoers unknown is justified. As always, all man-made tools, including Peer-to-Peer-networks, may or may not be used in a legitimate way.