GAS: Overloading a File Sharing Network as an Anonymizing System

Elias Athanasopoulos[1], Mema Roussopoulos[2,1], Kostas G. Anagnostakis[3], and Evangelos P. Markatos[1]

[1] Institute of Computer Science (ICS)
Foundation for Research & Technology Hellas (FORTH)
{elathan,mema,markatos}@ics.forth.gr
[2] Harvard University
mema@eecs.harvard.edu
[3] Institute for Infocomm Research, Singapore
kostas@i2r.a-star.edu.sg

Abstract

Anonymity is considered as a valuable property as far as everyday transactions in the Internet are concerned. Users care about their privacy and they seek for new ways to keep secret as much as of their personal information from third parties. Anonymizing systems exist nowadays that provide users with the technology, which is able to hide their origin when they use applications such as the World Wide Web or Instant Messaging. However, all these systems are vulnerable to a number of attacks and some of them may collapse under a low strength adversary. In this paper we explore anonymity from a different perspective. Instead of building a new anonymizing system, we try to overload an existing file sharing system, Gnutella, and use it for a different purpose. We develop a technique that transforms Gnutella as an Anonymizing System (GAS) for a single download from the World Wide Web.

Keywords: Security, Anonymity, P2P, Gnutella

1 Introduction

Anonymity is considered as a valuable property in both physical and digital life. Acting in an anonymous fashion may be considered suspicious, and in many cases the actor may not follow legal procedures. Apart from these ill cases, anonymity is considered the basic building block for someone who demands to keep her privacy. We are interested in cases, in which technology is used to preserve a user's private information from third parties.

David Chaum[7] in 1981 was the first to vision an electronic mail service that will operate in an anonymous fashion, using some intermediate MIX nodes. These intermediate MIX nodes decouple the correlation between the sender and the recipient; this correlation is the outcome of having the recipient contacting directly the electronic mail service. Since then, many systems that are especially

developed in order to serve as anonymizing systems have been published. These constructs may be used as means for having interactions with third parties without revealing the user's identity. More precisely, anonymity is usually required when two parties are having a transaction. For example a Web client is visiting a Web server. We distinguish between *sender* and *receiver* anonymity. In this paper we are dealing with sender anonymity; we develop techniques that allow a user to send an HTTP request to a Web server and receive the Web content, without revealing her identity.

Many anonymizng systems like Crowds[14], Tarzan[10], Cashmere[20] and MorphMix[15] have been designed theoretically and released as academic publications; some others, like Tor[11] and Freenet[8], have been deployed and are actively used. All these systems have been created from the ground up, with the explicit purpose of providing guarantees for a user's anonymity. Using, for example, Tor[11] a user may surf the World Wide Web without revealing her identity (her IP address) to the Web sites she visits.

In this paper, we do not build another anonymizing system. Instead, we use an existing system, Gnutella[1], whose prime goal is the exchange of files between users, as a mean of delivering a file from a Web site to a user in an anonymous fashion. We also present guidelines on how to use Gnutella as a covert communication channel.

The main contributions of this paper are the followings:

- We overload a file-sharing system for anonymous Web downloads.
- We use an existing system in a way, which was not included in the original implementors' intentions.
- We evaluate the strength of anonymity provided by our technique using metrics already proposed by academia.
- We present some ideas and guidelines on how to use the Gnutella system as a covert communication channel.

The rest of this paper is organized as follows. Section 2 highlights the basic technical details of the Gnutella architecture. This Section aims on making the reader, who has not further experience with Gnutella in the past, be able to follow the rest of the paper. In Section 3 we present the fundamental concepts of our techniques, which transform Gnutella into a means for anonymous Web downloads and in Section 4 we evaluate the strength of anonymity provided by Gnutella using well established metrics. In Section 5 we present some ideas on how to transform Gnutella into a covert communication channel. We present related efforts in Section 6 and we conclude in Section 7.

2 Gnutella Architecture

Gnutella[1] is an open system targeting file sharing. It promotes the peer-to-peer paradigm and it is purely a decentralized distributed system having millions of concurrent participants. This section describes fundamental concepts of the Gnutella architecture.

2.1 Generic View

Gnutella is built on a two layer random graph topology. The core layer composed by some thousands of peers that are involved in routing of messages. These peers are also called Ultrapeers. Each Ultrapeer maintains approximately 30 connections with other Ultrapeers and approximately 30 connections with peers of the second layer, which are called Leaves. Leaves are not involved in routing; they send and receive messages via their Ultrapeers. Each Leaf is connected into approximately 3 Ultrapeers.

More information about the Gnutella topology and peer distribution's characteristics can be found in [18].

Each basic operation in the Gnutella system is carried out by constructing messages and routing them through the overlay. The basic lookup operation uses the flooding algorithm to query the overlay. A peer constructs a message that embeds search criteria relative to the file it is looking for, and it forwards the query message to its neighbors. Its neighbors further forward the message to their neighbors and so on. Along the path the original message is routed, every peer is free to answer to the query, by constructing a message with possible results relative to the search criteria and its identity; namely its IP address and a Port number. This message is routed back to the original peer that issued the lookup operation following the reverse of the path taken by the query.

Finally, if a peer is satisfied with the search results, it connects to the peer using the identity embedded in the search results message and it downloads the file using the HTTP protocol.

2.2 Gnutella as a Web Download Platform

It is shown in [5] that with the current Gnutella architecture it is possible for a Web server to become advertised in Gnutella search results messages. Furthermore in [5] the authors have developed techniques that trick Gnutella peers into downloading a specific file from a Web server.

Shortly, in [5] the authors managed to misuse the lookup procedure of the Gnutella protocol, in order to force peers to download files from third parties, and especially, as it is illustrated in detail in [5], from Web servers. The main idea, is to insert a malicious peer in the Gnutella system that answers query messages with responses, which instead of containing the identity of the malicious peer, they contain the identity of a third party; a Web server. This is feasible, because Gnutella is completely decentralized and there is no way to distinguish between a message contains authentic information or a message that is specially crafted to contain fake information.

Acting as described above has the effect of tricking Gnutella peers to try to download files from a Web server. Furthermore, the authors in [5] have managed not only to trick Gnutella peers to request a file from Web server, but they illustrate that it is possible to trick Gnutella peers to *actually download a specific file* from a Web server. This can be achieved by embedding special crafted filenames in query responses that take advantage of some HTTP characteristics, so as to

force a Web server to serve a specific file upon accepting a download request from a Gnutella peer.

The authors' intentions in [5] was to cause a distributed denial of service attack to a Web server by tricking Gnutella peers to massively request a file from a Web server. In this paper, we build upon this idea to trick a few peers to fetch a file from a Web server inside Gnutella and then make a user able to download the file from Gnutella instead of requesting it from the Web server. In this way, a user can download a file from a Web server in an anonymous fashion; without coming in direct contact with the Web server.

2.3 HOPs Spoofing

A vital characteristic of the Gnutella system is that you can never tell if a message originates from a peer or the latter simple routes the message on behalf of another one. More precisely, each message of the Gnutella protocol has a TTL (Time To Live) and HOPs field. Each peer is responsible for increasing the HOPs field and decreasing the TTL field upon a route operation. When the TTL field reaches the zero value, the peer that received the message is responsible for dropping the message from the system. The interesting part is that each peer is free to create messages with spoofed TTL and HOPs field. In this fashion, a peer can inject a new message into the system, but, by spoofing the HOPs and TTL field, the peer can pretend that the message is routed and not created by itself.

3 GAS Architecture

In this section we analyze in detail the GAS architecture. In Table 1 we list the meaning of symbols used frequently in the current and following sections.

3.1 Overview

GAS' goal is to transfer a file from a Web Server to a computer machine in an anonymous fashion. Strictly, what we are trying to achieve is:

> **GAS' Goal**: Transfer a file, F, from a Web Server, W, to a computer machine, C, that never comes in a direct contact with W, nor with another computer, which is able to prove with a great probability that C was in contact with W in order to retrieve the particular file.

In order to achieve GAS' goal we use the real-world file-sharing system Gnutella. The whole GAS algorithm is divided into two separate phases: the *FETCH* and the *MIX* phase. The *FETCH* phase transfers F from W to some peers in the Gnutella system and the *MIX* phase populates F in the Gnutella system. We explore the two different phases of GAS in the following paragraphs.

Symbol	Explanation
C	Computer, GAS user, the user who desires the anonymity
F	The desired file located in a Web sever
W	The Web server that hosts the desired file
$FETCH$	Initial phase of GAS (a file is transfered from the Web server to Gnutella)
MIX	Second phase of GAS (a file hosted by Gnutella peers is further propagated to the Gnutella system)
d	Degree of Anonymity (anonymity metric)
H_M	Maximum entropy of the system
$H(X)$	Entropy of the system after the adversary's attack
N	Peers that compose the set of the $FETCH$ phase
M_i	Peers that compose the set of the ith MIX phase
A_c	Peers which are controlled by the adversary

Table 1. Symbols which are used frequently in the description of the GAS architecture and evaluation.

3.2 FETCH Phase

The *FETCH* phase is the initial part of GAS. The main goal of the *FETCH* phase is to transfer F from W to Gnutella. This must be done, without having a direct contact of C with W. Thus we use the techniques illustrated in [5]. In [5] the authors present a technique to use Gnutella as a Denial of Service attack platform against Web Servers. More precisely the authors have developed a technique that tricks Gnutella peers to download a file from a third party Web server. If a large population of Gnutella peers is tricked to download from the Web server, the Web server is faced with a flash crowd event and eventually becomes a victim of a distributed Denial of Service attack. In GAS we employee the same idea, but we use it in a controlled way, so as not to cause any harm in Web servers.

GAS uses a special modified peer, C, as it is done in [5], in order to advertise F, which is served by W, in Gnutella, so that some Gnutella peers proceed and download the file.

During the *FETCH* phase we assume that N Gnutella peers will have downloaded F from W. Note, that there is no way - up to this point - for any external observer to associate with any information C with W. Since C is connected to the Gnutella system and feeds it with fake results, that embed the identity of W.

According to the experiments illustrated in [5] a modified peer which issued 10,000 fake results managed to trick 133 Gnutella peers in 10 minutes of working time. More experiments can be found in [5].

Up to this point, we have managed anonymously to force Gnutella to download F from W. We could use C to query for F the Gnutella system, using some keywords that are relevant to F's name and then download F from one peer that belongs to the set of peers N that have fetched F from W. In this fashion, C has

managed to retrieve F without coming in direct contact with W but only with some tricked Gnutella peers. However, as we further investigate in Section 4, we assume that an adversary against GAS has managed to control a peer, A, which belongs to the N set of Gnutella peers. This peer may associate C with W since A and W may cooperate in order to attack GAS. In order to deal with this case we employ the *MIX* phase.

3.3 MIX Phase

The *MIX* phase aims on populating F in the Gnutella system in a chaotic way. That is, C tries to further trick other Gnutella peers to download F from the N peers that were tricked during the *FETCH* phase.

In order to achieve this C needs to have a list of IP addresses that potentially map to tricked peers belonging to the N set. As it is explained in detail in [5], in order for a malicious peer to trick a peer to download a file served by a Web Server or another Gnutella peer, the malicious one must know in advance the IP address of the victim, in order to embed it in fake QueryHits (reply messages to Query messages). It is hard for C to know exactly which peers were tricked to download F from W, since the download process via Gnutella involves in great extent the human factor. However, C can try to use as victims all the peers which it sent fake responses during the *FETCH* phase. This process requires C to know for which peer it generates a fake response when it receives a Query message. In the original Gnutella specification a Query message does not embed the identity (IP address and Port number) of the initial querier. During the last few years, an extension has been introduced in order to promote direct delivery of responses to requesting peers via UDP[4]. Peers that support this extension include their identity in their Query messages. We performed measurements, using a modern Gnutella client[2], in order to find out the ratio of popularity of this extension in current Gnutella. In other words, we measured how many Query messages embed Gnutella identities. In Table 2 we list the results. The majority of Query messages in current Gnutella embed identities of Gnutella peers. Thus, C can generate fake responses only for Query messages, which embed a Gnutella identity and keep the identity in a list. These collected identities may potentially take part in the N set.

Having collected an identity list of potentially tricked peers, the N set, C may proceed and further trick other Gnutella peers to download F from peers belonging to N set. These new tricked peers compose the M_1 set.

The *MIX* phase may be applied i times in order to gain stronger anonymity. Finally, C may query the peers composing the M_{i-1} set and download F from them. That is C will be - artificially - part of the final M_i set.

4 GAS Evaluation

There are several attempts to quantify the anonymity provided by an anonymizing system. Most recent papers [9], [16] and [19] use the notion of information

Queries Received	OOB Queries	Percentage
1,000	751	75%
2,000	1,536	77%
3,000	2,183	73%
4,000	2,840	71%
5,000	3,498	77%
10,000	6,617	66%

Table 2. OOB Queries percentage, measured using a modern Gnutella client [2]. OOB Query messages embed the identity (IP Address and Port number) of the peer that issued the Query.

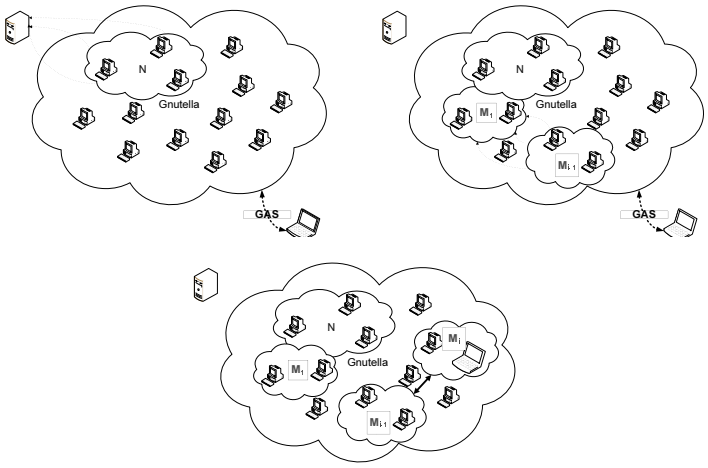

Fig. 1. The different phases of the GAS architecture. During the *FETCH* phase N Gnutella peers are tricked to download a file from the Web Server. During the *MIX* phases, the file is further populated in the Gnutella system (in M_i sets). Finally, the user of GAS participates in the final M_i set and downloads the requested file from peers in the M_{i-1} set.

entropy to measure the information which leaks from the system and can be used by an adversary in order to degrade the anonymity provided by the system. We will use [9] in order to measure the *degree of anonymity* provided by GAS. This theoretic metric depicts the effort required from an adversary to be able to identify with great probability a sender from a set of possible senders.

In [9] the authors have illustrated the degree of anonymity of known systems such as Crowds[14] and Onion Routing[11] and, thus, using the same methodology GAS can be compared with the above systems.

Before we proceed, in the rest of this evaluation the term *anonymity* follows the precise definition given by [13]: *the state of being not identifiable within a set of subjects, the anonymity set.*

The degree of anonymity as expressed in [9] is related to *sender anonymity*. Of course, as the authors mention in the paper, *recipient anonymity* can also be modeled using the same concept. It is vital to understand that, in contrast to other anonymizing systems, GAS introduces the notion of separate phases, the FETCH and a series of possible MIX phases. In order for complete Web transaction to take place using GAS, C becomes a *sender* and a *recipient*. It is a pure sender during the FETCH phase and the possible MIX phases, but in the final stage, when the actual download is performed from the M_i set, C is a recipient. Thus, we have to evaluate the degree of anonymity of GAS in two separated phases: one, in which the user acts like a sender and one, in which the user acts like a recipient.

4.1 Degree of Anonymity

As the authors state in [9] the degree of anonymity is expressed as:

$$d = \frac{H(X)}{H_M}, \tag{1}$$

where

$$H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

is denoted as the entropy of the system after the attack has taken place, while

$$H_M = \log_2(N),$$

is denoted as the maximum entropy of the system, when N is the size of the anonymity set that includes the number of legitimate senders (or recipients).

In the above, we denote with p_i the probability mass function $p_i = Pr(X = i)$, where i represents each possible value the discrete random value X may take. Specifically, each i corresponds to an element of the anonymity set (a sender or a recipient). If we have a system with i possible senders, an adversary assigns a probability p_i to each one of the set.

4.2 Degree of Anonymity of GAS

In order to estimate the degree of anonymity in GAS we have to define the adversary model. We believe that a realistic adversary model is an *active attack*. That is, the attacker has under its full control A_c nodes that take part in the GAS *MIX* phase.

The maximum entropy of GAS can be measured if we define the anonymity set that includes the nodes of Gnutella that take part in the process of the file downloading. Let G_p to be the complete population of Gnutella nodes. The nodes that take part in GAS are

$$N + \sum_{i=1}^{m} M_i,$$

assuming we have m *MIX* phases. It is always

$$N + \sum_{i=1}^{m} M_i \leq G_p.$$

The maximum entropy of GAS is:

$$H_M = \log_2(N + \sum_{i=1}^{m} M_i). \tag{2}$$

Now, each node in the A_c may come in contact with:

- the Web server, W, and possibly with a Gnutella peer that belongs to the M_1 set, if it is part of the N set,
- another Gnutella peer that belongs to M_{i+1} if it is part in the M_i set.

The adversary must assign probabilities to each node it comes in contact with. Assuming, that the GAS user is hidden in a Gnutella subset, i.e. it may not be distinguished in a trivial way from other Gnutella nodes that take part in an M_i set, the probability is uniform and depends on the requests recorded from the adversary. Thus, the adversary assigns $p_i = \frac{1}{R_{in,i}}$, where $R_{in,i}$ denotes the number of incoming requests of a node controlled by the adversary which is in an M_i set. It is always $R_{in,i} \leq M_{i+1}$.

There is only one case, where the adversary may spot the GAS user with great probability. Assuming we have m *MIX* phases, if the adversary manages to control all nodes in M_{m-1} set and the final set M_m includes only the GAS user, then the identity of the GAS user is fully revealed. We argue that, although this is possible, it may be adjusted by the GAS user, so that each M_i set is quite dense. This is manageable, since the GAS user constructs the sets, by tricking other Gnutella peers. In addition, it is difficult for the adversary to know in which *MIX* set she belongs to - it requires the effort of having at least one node controlled by the adversary in each *MIX* set.

Thus, in the general case and assuming that $R_{in,i} = M_{i+1} - A_c$ (this, simply, means that a node in the M_{i+1} set is forced to generate requests to each node of the M_i set, excluding nodes controlled by the adversary), the probability assigned by the attacker to each GAS participant is $p_i = \frac{1}{M_{i+1} - A_c}$. The entropy for a given set k will be:

$$H(X) = -\sum_{i=1}^{M_{k+1}-A_c} \frac{1}{M_{k+1} - A_c} \log_2(\frac{1}{M_{k+1} - A_c}). \tag{3}$$

In order to calculate the entropy of the whole system, we need to sum up all the probabilities for all *MIX* phases.

4.3 Degree of Anonymity in a MIX phase

Assuming we have only one MIX phase the degree of anonymity is:

$$d = -\frac{\sum_{i=1}^{M-A_c} \frac{1}{M-A_c} \log_2(\frac{1}{M-A_c})}{log_2(M)} = \frac{log_2(M-A_c)}{log_2 M}. \quad (4)$$

Since, we are interested in the degree of anonymity of a MIX phase, we excluded from H_M the N nodes that take part in the initial $FETCH$ phase.

In Figure 2 we depict the degree of anonymity for a given MIX phase for various set's populations.

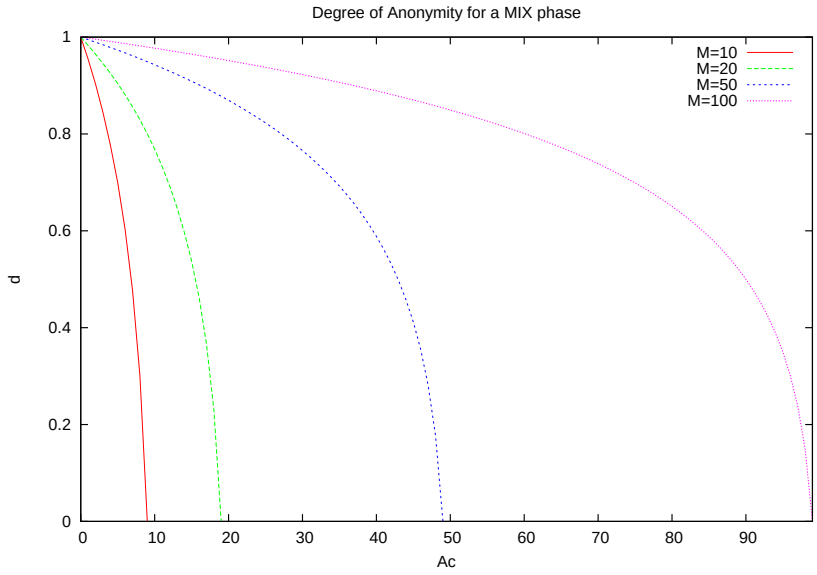

Fig. 2. Degree of Anonymity for a MIX phase for various set's populations. The degree of anonymity is expressed in equation (4). Initially the degree of anonymity provided by the system is equal to 1 (complete anonymity). As the adversary injects more nodes in the system, the degree of anonymity is reduced. If the adversary manages to substitute all nodes with nodes controlled by her, then the anonymity degrades to zero.

4.4 Degree of Anonymity of a series of MIX phases

In order to calculate the degree of anonymity of the whole system, we have to sum up all the MIX sets and extend equation (4) to include the nodes that compose all the MIX phases. It is trivial to observe that equation (4) will not differ with the equivalent of a series of MIX phases, since in order to calculate

the entropy of a series of MIX phases we just sum up all senders (all the M_i sets) and subtract the nodes which are controlled by the adversary. Thus:

$$d = -\frac{\sum_{i=1}^{M_k - A_c} \frac{1}{M_k - A_c} \log_2(\frac{1}{M_k - A_c})}{log_2(M_k)} = \frac{log_2(M_k - A_c)}{log_2 M_k}, \quad (5)$$

where now with M_k we denote the sum of the nodes of all the MIX phases. We omit the $\sum$ symbol for readability reasons.

4.5 The benefit of having multiple MIX phases

Comparing equations (4) and (5) reveals that there is no benefit of having multiple MIX phases in terms of the *degree of anonymity* metric. In other words, the degree of anonymity, according to the above model, depends only in the percentage of nodes controlled by the adversary over the nodes that are tricked by the GAS user and serve as possible senders. To give it with numbers, two MIX phases, M_1 and M_2, with A_{c1} and A_{c2} nodes controlled by the adversary have the same degree of anonymity with one MIX phase, M', having A'_c nodes controlled by the adversary, if $M' = M_1 + M_2$ and $A'_c = A_{c1} + A_{c2}$.

This seems a little bit contradictory. Why, then do we need multiple MIX phases? The answer is that having multiple MIX phases forces the attacker to spread the controlled nodes to all MIX sets if she wants to calculate correctly the degree of anonymity of the last MIX set. Recall that the GAS user performs a download - and this action is the only action which qualifies as evidence that someone is utilizing GAS - only as part of the last MIX set.

By having only one MIX phase, the adversary may assign p_i probabilities to all the nodes taking part in the MIX phase. By having multiple MIX phases, the adversary must distinguish nodes that take part in different MIX phases and focus on the nodes of the last MIX set. This requires the effort of having at least one attacker in each of the MIX sets.

The intuition of having multiple MIX phases is the same as having multiple chaumian $MIXes$ in a $MIXnet$ [7].

4.6 Comparison of GAS and other ASs

In [9] there is a similar evaluation with the above one regarding the degree of anonymity in existing anonymizing systems. It is interesting to observe that the degree of anonymity of GAS is equal to the degree of anonymity provided by Onion Routing [11]. However, there are some properties of GAS that differentiate [4] it from Onion Routing. In terms of advantages, GAS has the following properties:

[4] The degree of anonymity is a well defined and accepted metric in order to measure the anonymity provided by a system. However, some practical characteristics of a system make it ideal for performing some tasks in an anonymous fashion than another one, even if both systems have exactly the same degree of anonymity.

– The sender sets of GAS can have much more entries than a typical sender set of Onion Routing,
– The adversary must inject many nodes to control a substantial number of nodes in the system, so as to be able to perform a realistic attack.

In terms of disadvantages:

– GAS does not have as fine-grained controls control as Onion Routing,
– GAS can not be used for real-time Web Browsing, but for a simple download from WWW,
– GAS uses unencrypted communications.

5 Using GAS as a covert communication channel

Besides using Gnutella as a platform for anonymous downloads via the WWW, Gnutella may also transformed into a covert communication channel. For example, two parties that want to communicate anonymously via Gnutella can exchange Query and QueryHit messages. These messages can be also encrypted. Below we list a numerous ways of using some of the Gnutella internals to transform the system into a covert communication channel.

5.1 Using the Gnutella lookup procedure

A peer may flood a query with encrypted payload[5] in Gnutella. A colluding peer, which is located in the initial peer's horizon may answer the Query with a response with encrypted payload[6]. The two colluding peers have managed to:

– Exchange information using the lookup mechanism of Gnutella.
– Exchange information using encrypted messages.
– Exchange information without coming in direct contact with each other, but using Gnutella as a means to route the information from one party to the other.

The Query message, since it is flooded in the system, will be received from many peers, as well as the response in the Query message will be routed back to the initial peer through a series of Gnutella peers. Since both payloads of the Query and the Query response are encrypted, it is unlikely that an adversary that has injected its nodes in the system can perform a Man in the Middle attack and reveal the contents of the messages. The two colluding peers may also exchange secret keys occasionally again using the lookup operation of the Gnutella system.

[5] We consider the search criteria of a Query message as the payload of a Gnutella query

[6] We consider the search results of a Query response message as the payload of a Gnutella Query hit.

5.2 Using the Gnutella message tagging

Apart from taking advantage of Gnutella's lookup procedure to exchange private information there are more elaborate methods for using the Gnutella system as a covert channel. A lot of Gnutella messages contain unused bits; bits that will never used by a normal Gnutella client. Two colluding peers may inject the information they want to exchange in these unused bits. In order to illustrate a more concrete example, consider that each Gnutella message is tagged with a 16-byte GUID[3]. Assuming that a GUID tag is totally random, two colluding parties may arrange to use some part of the 128-bit GUID tag as an information carrier. All normal Gnutella clients will treat this portion of the GUID tag as a random subportion of the complete identifier, but the colluding peers will treat specially this portion of information according to their prior arrangement.

5.3 Using the Gnutella PING-PONG exchange

A less efficient, in terms of flexibility, technique is to use the PING-PONG exchange mechanism to transmit and receive secret information. Although, this mechanism is not widely used in the current Gnutella system, since it produces network traffic overhead, it is still possible for peers to use this mechanism to discover new hosts. When a peer does not have many entries in its host cache and it is in need of discovering new Gnutella peers, it broadcasts (using again the flooding process) a Gnutella PING packet. In response, it receives Gnutella PONGs from active Gnutella peers. These PONG messages embed the peers' identities (IP address and Port number) as well as some statistics in regards to the data files they share. Again, two colluding peers may encode secret information in these PING-PONG messages. For example a peer may encode the message it wants to transmit in a PONG packet. Peers that passively monitor the PONG traffic will try to connect to a non existing IP address and Port number, since it is unlikely that the special encoded message will map to a valid Gnutella IP address and Port.

The profound property of Gnutella that makes all the above easily achievable is the lack of a central mechanism to verify if a message is authentic or fake. Since every Gnutella participant may inject messages in the system that seem to originate from another node and not by itself, then it is feasible to also embed in these messages secret information, which can only be interpreted in a meaningful way by another colluding node (see HOPs spoofing in Section 2.3).

6 Related Work

Since David Chaum introduced the term of an anonymous and untraceable electronic mail in 1981 [7] a lot of research has been taken place in the academic and industrial field. Nowadays, there are plenty of anonymzing systems, such as Crowds [14], Tarzan [10], MorphMix [15], Freenet [8], Cashmere [20] and Tor [11]. Each system tries to provide anonymous communication by routing messages between a sender and a receiver through nodes that try to decouple any

relation between the two communicating parties. It is worth mentioning that all these systems have certain advantages and disadvantages, and thus this is an active field for further research.

As far as GAS is concerned the two key properties that differentiate it from current anonymizng systems are that (a) GAS is built over an existing infrastructure that was not designed to provide any mean of anonymous communication and (b) GAS is built over a large set of nodes that can be used as relayers in order to provide anonymous communication.

As far as the first property is concerned, there has been an attempt for utilizing the World Wide Web as a covert channel [6] in order to provide anonymous communication. Beyond that, there is no similar work in exploiting existing systems, designed for other purposes than anonymous communications, for anonymity purposes, as far as we know.

As far as the second property is concerned, most of current anonymizing systems rely on a small set of nodes that in most cases [10, 11, 15, 20] full knowledge of the system is required, and thus there are scalability issues. Anonymizing systems such as P^5 [17] and Salsa [12] have been designed in order to provide scalable anonymous communications. But, again it is hard to implement a dedicated system for anonymous communication that hosts a few millions of nodes.

7 Concluding Remarks

In this paper we presented GAS (Gnutella as Anonymizing System), which transforms the open file sharing system, Gnutella, to a platform for performing file downloads from the World Wide Web in an anonymous fashion.

We furthermore evaluated GAS with already accepted scientific metrics, such as the degree of anonymity [9]. We compared GAS using the metric of degree of anonymity with other anonymizing systems and showed that GAS has the same degree of anonymity with Onion Routing. We presented crucial properties of GAS which differentiate it from Onion Routing in various aspects.

To conclude, we believe that this paper is the first attempt to use an already existing system, which has not been initially designed as an anonymizing system, for anonymous communication. The fact that GAS is based in Gnutella, which has not been designed for anonymity purposes, makes it less controllable and unsuitable for real-time communications compared to other practical implementation of anonymizing systems, like Tor [11] for example. But, on the other hand, the increasingly popularity of Gnutella, gives GAS, potentially, a very large anonymity set composed by millions of nodes, which, as far as we know, has never been accomplished by any other anonymizing system.

In terms of the degree of anonymity metric and according to equation (5) having a large anonymity set requires from the attacker to inject more attacking nodes in the set in order to reduce the metric, and thus degrade the anonymity provided.

We believe that GAS is not suitable for real-time Web Surfing, but for a single Web download, it is up to the GAS user to trick a huge base of Gnutella peers and form a huge anonymity set that will hide her activities inside.

Last but not least, we listed some possible usages of Gnutella as a covert communication channel.

8 Acknowledgments

We thank the anonymous reviewers and George Danezis (K.U.Leuven) for his valuable feedback in various aspects of GAS. This work was supported in part by project SecSPeer (GGET USA-031), funded in part by the Greek Secretariat for Research and Technology and by the CoreGrid Network of Excellence. Elias Athanasopoulos, Mema Roussopoulos and Evangelos P. Markatos are also with the University of Crete.

References

1. Gnutella protocol. http://rfc-gnutella.sourceforge.net/.
2. Gtk-gnutella servent. http://gtk-gnutella.sourceforge.net.
3. GUID Specification. http://en.wikipedia.org/wiki/Guid.
4. OOB Specification. http://gtk-gnutella.asselman.com/gtk-gnutella-current/doc/gnutella/out-of-band.
5. Elias Athanasopoulos, Kostas G. Anagnostakis, and Evangelos P. Markatos. Misusing Unstructured P2P Systems to Perform DoS Attacks: The Network That Never Forgets. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 130–145, 2006.
6. Matthias Bauer. New Covert Channels in HTTP: Adding Unwitting Web Browsers to Anonymity Sets. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, October 2003.
7. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.
8. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, July 2000.
9. Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.
10. Michael J. Freedman and Robert Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.
11. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.
12. Arjun Nambiar and Matthew Wright. Salsa: A Structured Approach to Large-Scale Anonymity. In *Proceedings of CCS 2006*, October 2006.

13. Andreas Pfitzmann and Marit Hansen. Anonymity, Unobservability, and Pseudonymity: A Consolidated Proposal for Terminology. Draft, July 2000.
14. Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.
15. Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.
16. Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.
17. Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5: A protocol for scalable anonymous communication. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002.
18. Daniel Stutzbach and Reza Rejaie. Characterizing the two-tier gnutella topology. *SIGMETRICS Perform. Eval. Rev.*, 33(1):402–403, 2005.
19. Gergely Tóth, Zoltán Hornák, and Ferenc Vajda. Measuring anonymity revisited. In Sanna Liimatainen and Teemupekka Virtanen, editors, *Proceedings of the Ninth Nordic Workshop on Secure IT Systems*, pages 85–90, Espoo, Finland, November 2004.
20. Li Zhuang, Feng Zhou, Ben Y. Zhao, and Antony Rowstron. Cashmere: Resilient Anonymous Routing. In *Proc. of NSDI*, Boston, MA, May 2005. ACM/USENIX.