GNUnet – anonymity for free

Krista Bennett, Christian Grothoff, Tzvetan Horozov, and Ioana Patrascu

Department of Computer Sciences, Purdue University
{klb,grothoff,horozov,patrascu}@cs.purdue.edu
http://gecko.cs.purdue.edu/GNUnet/

Abstract. This paper describes how anonymity is achieved in GNUnet, a framework for anonymous distributed and secure networking.
We describe a new scheme for anonymous transfer of data which achieves better anonymity guarantees than traditional indirection schemes and is more efficient. While the building blocks of our technique used to achieve anonymity are identical to previous work, we offer a new perspective on how to perceive and measure anonymity.
THIS PAPER IS NOT FINISHED AND IS PROVIDED ONLY AS A REFERENCE FOR GNUNET DEVELOPERS AND USERS AT THE MOMENT.

1 Introduction

Large public networks present a challenge to any system that needs to provide security guarantees to its users. This is especially true because the widely used TCP/IP protocols [6] lack mechanisms for authentication and confidentiality [1]. The primary applications for distributed applications are computation and data transfer. In this paper, we focus on anonymous data transfer. The traditional methods of transferring files over networks have serious drawbacks which are partially caused by the underlying network technology. Some of these drawbacks include:

- Communications are not anonymous. In general, the sender, the receiver and intermediaries (e.g. routers) are exposed and easily observed.
- Data transfers can be intercepted and manipulated. Application-level protocols such as ssh [13] and ssl [7] address this issue. For this paper, we will assume that data transfers between participating hosts are safely encrypted and that participating hosts are properly authenticated.

In this paper we first describe our assumptions and the requirements for anonymity that we try to fulfill. Section 3 describes related work. We then describe how anonymity is achieved in GNUnet, thus allowing participants to avoid liability. Finally, parts of the GNUnet implementation relating to anonymity are discussed.

2 Basic Assumptions and Definitions

In this paper we assume that queries and content are sent encrypted though the network such that neither the node sending back the reply nor any intermediaries can decrypt them. This is guaranteed by a scheme described in [2].

2.1 Assumptions about the Network

In this section we describe the assumptions made about the underlying network (GNUnet) that is supposed to facilitate the anonymous communication. While GNUnet is an open network and every host on the internet could join it, we distinguish between adversaries that participate and adversaries that merely observe.

We assume that communication between the hosts in the network is confidential, such that no hosts outside the network can even observe the meta-data that identifies the type of information (e.g. query or content) that flows through the network. All communications between hosts are also authenticated. Availability is guarded by an accounting scheme that is based on this link-to-link authentication and that does not require end-to-end knowledge about transactions.

2.2 Deniability

A weak form of a man-in-the middle attack occurs if a malicious host joins the network (thereby forming a link). The attacker could also intercept an exchange of public keys and join the network claiming false IPs. Because identities in GNUnet are public keys, both approaches are equivalent in their impact on the network.

A very powerful adversary could prevent a node from connecting to any node that is not controlled by the attacker. If the attacker is able to ensure that the victim is only able to connect to nodes controlled by the adversary, the victim is obviously unable to communicate anonymously as every communication that does not originate from the adversary must originate from the victim. The best protection that we can offer in this situation is to make it hard for the adversary to find out what the victim was trying to do. We call this property *deniability*.

The argument above demonstrates that in order to guarantee any form of anonymity, we must assume that every node knows and is able to communicate with at least one other node that is not controlled by a malicious adversary. It is sufficient to assume that this node is controlled by an adversary that does not collaborate with the adversary controlling all the other connections.

Given this system, an adversary can see any communication between two nodes, but not decrypt that communication. An adversary can join the network and then observe and change parts of the meta-information that flows through the network. In particular, an adversary can determine if a query or content was sent. The only restriction on the adversary (other than not being able to break cryptographic primitives) is that it does not control all the hosts that a node in the network communicates with.

2.3 Anonymity

We say that a communication is anonymous if no adversary can say **with certainty** which two partners were the ultimate source and the ultimate destination of the communication. The adversary may also take the place of the source or

the destination. Anonymity can be *measured* by determining the *probability* that a certain host was the source (or destination) of a communication. We reject as frivolous any categorization of degrees of anonymity. The question of whether or not a system is *sufficiently* anonymous for a specific application depends entirely on the purpose.

It is important to note that we put the **burden of proof** on the adversary. The adversary must prove that a communication originated from us. On the other hand, if the legal system requires us to disprove being the origin, anonymity is essentially outlawed.[1]

2.4 Efficiency

The efficiency of an anonymous network depends on the computational overhead, the bandwidth required (relative to the amount of data transferred) and the number of communications that must be performed. The number of communications gives an estimate of the delay for an operation.

The computational overhead is dominated by the encryption time in all systems and is not discussed further in this paper. Systems that do not rely on confidential or authenticated communications between the nodes are obviously open to sniffing attacks and are thereby not suitable for the problem at hand.

3 Related Work

Traditionally, indirection has frequently been used in order to achieve anonymity [4, 14, 10]. Forwarding queries from other hosts allows users to deny that a packet originated from a particular host (assuming the adversary was not able to perform full-traffic analysis). The principle of indirection applies to queries as well as to replies.

Indirecting all communications is very costly. For example, in Freenet [5], the number of indirections is determined by the length of the search path that the query takes until a host that has the content is found. If the search path has length l, there are, thus, l transfers of the content. If the content is large enough, the traffic overhead is then $(l-1) \cdot s$ where s is the size of the content.

Other systems, like Crowds [11], allow the user to set the number l of indirections that the system should aim for. While the traffic overhead is again $(l-1) \cdot s$, the l can be tuned. The assumption here is that a larger n increases the anonymity of the system.

In Crowds the authors describe a network where n hosts indirect communications with a probability p_f. The authors then argue about the probability that a collaborating node receives a communication from the node that actually sent the request.

[1] Anonymity would require us to prove that a communication did not originate from us. Because the communication did originate from us, we would have to construct a false proof. The judge would, obviously, reject any kind of proof that could be wrong.

The authors of Crowds assume that all nodes are equally active and thus equally suspicious. Even if the adversary has only c nodes under control, traffic analysis may give much better data on which node is responsible for the query – even under the assumption that traffic between the $n - c$ non-malicious nodes cannot be decrypted. Sending noise to make the traffic analysis harder is not discussed and would, of course, increase the network load beyond $(l - 1) \cdot s$.

All systems providing anonymity that we know of are based on these principles and suffer from the same basic drawbacks as described above. The overhead is usually on the order of $(l - 1) \cdot s$.

4 Anonymity in GNUnet

In this section, we describe how anonymity is achieved in GNUnet. In order to be able to evaluate the anonymity guarantees that GNUnet provides, we first describe a new perspective on how indirecting communications result in anonymity. Then we describe the scheme deployed in GNUnet and discuss its guarantees and efficiency.

4.1 Hiding the Initiator of Activity

Consider the following scenario illustrated in figure 1. In this scenario a node receives two queries and sends three: In this picture, the two nodes that send their

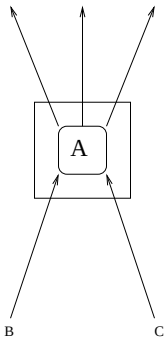

Fig. 1. Hiding

queries are exposed. The node A can correlate these nodes with their queries; a traffic analysis reveals that B and C sent a query. If A is allowed to forward a query twice, traffic analysis alone cannot reveal if A sent a new query or was merely indirecting.

In this sense, indirections do not hide the senders; instead, indirections obfuscate what the node that is indirecting is doing. No scheme that tries to achieve

anonymity on an observable network can hide the fact that a node is *participating*. The best a scheme can do is guarantee that no adversary can distinguish activity that a node initiates from mere participation in the protocol.

As the above example demonstrates, a node can hide its activities by indirecting activities from other nodes.

4.2 Anonymity Guarantees

In order to answer the question of how strong the anonymity guarantees are that indirection can provide, some additional constraints must be considered. The more traffic a node A sends into the network, the more indirected traffic is needed by A to divert from the traffic A generated itself. If node A injects n queries into the system and indirects m queries from other users, then a passive adversary can guess with a probability of $\frac{n}{n+m}$ that a query originated from that user.

If the adversary uses timing analysis, it is possible that the adversary may be able to exclude certain queries that were indirected a long time ago. "Long" here depends on the potential delay that a query may commonly face in a node. Nodes can delay queries for random amounts of time in order to make this timing analysis harder. Delays for excessive amounts of time make the query useless and indirecting it equivalent to producing noise (with the exception that other nodes will not perceive it as such).

Queries that originated from an active adversary and were received by the adversary can also not be counted for m, as the adversary knows that these queries did not originate from A. As we assumed at the beginning, every node always interacts with at least one non-malicious node and thus is able to receive queries that do not originate from the adversary.

4.3 Malicious Hosts Improve Performance

Suppose a malicious host B in GNUnet does not change the sender address A of a query to its own address, but preserves the original address. In that case, the recipient C of that indirection can determine that B indirected the query, and C can eventually send the reply directly to A (see figure 2).

Notice, that while A and C now know each other, they can not be certain that the query originated from A or that the reply originated from C. Both A and C could just be nodes that obey the protocol and indirected the query (or the reply). Because the anonymity of A and C depends on how many packets they indirect for others, B did not damage their protection. On the other hand, B did damage its own protection. C is now able to tell that this particular query did not really originate from B; thus, C now has a higher chance of guessing which traffic actually orginated from B.

The malicious behavior from B did not cost A or C anything. However, because the reply is not indirected via B, the total amount of traffic that was produced has been reduced thus allowing this "malicious" behavior to be used to improve performance.

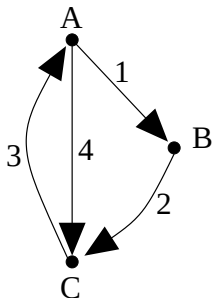

Fig. 2. Indirecting Replies

While this technique represents an improvement the performance gain could be even higher. Because A and C needed to communicate, A may decide to send the next query directly to C. If A is likely to send many related queries (related in the sense that the replies are likely to be on the same host), it is reasonable to assume that C will often be closer to the location of the document than B is.[2] This way, the number of hops between A and the content is decreased, thus speeding up the download process even further.

Let us suppose B is indirecting m queries and sending n new queries for its own user. As stated above, this would yield a probability of $\frac{m}{m+n}$ that any given query originates from B. If m is sufficiently large compared to n, this security may not be required by B. Indirecting m queries and m replies causes a great deal of work for B. If B chooses not to indirect k queries, and, instead keeps the original sender, the probability is increased to $\frac{m}{m+n-k}$.

5 Implementation

The implementation chosen in GNUnet is based on the philosophy outlined above. An implementation is available on our website

`http://www.gnu.org/software/GNUnet/`.

5.1 Joining the Network

A node that wants to join the network must know at least one public key (and Internet address) of another trusted node. The other node must be *trusted* in the sense that it is not a strong attacker that advertises a GNUnet network that is entirely under the adversary's control (in that case, the adversary could keep track of what the new node is doing). If the new node has several initial public

[2] The encoding of content in GNUnet [2] requires many related queries before a download of a single file can be completed. In other systems, queries may be related less often.

keys of other nodes, it is sufficient if one of these does not collaborate with the adversary.

Each node in GNUnet has an RSA key pair. The nodes use these keys to exchange 128-bit session keys that are used to establish a link-encryption infrastructure between the nodes. We are using Blowfish for the symmetric cipher. Nodes periodically sign their current Internet address (together with a timestamp) and propagate packets with public keys and Internet addresses. Except for the initial exchange of public keys that occurs when a node joins, this exchange of public keys can also use the encrypted channels.[3]

5.2 Queries and Replies

Nodes indirect queries and can thereby hide the queries originating from themselves. Every node uses a combination of network load (and other factors that are internal to the node) to determine how often to indirect queries. If the general network load is high, then the node indirects fewer queries, assuming that its own traffic is already well hidden. If the network load is low, then more queries are indirected.

Several queries are usually sent out in a group, potentially with other data. Grouping several queries to a larger packet introduces delays and decreases the per-query overhead. Encrypted packets containing queries can also not be distinguished from packets containing other data because the grouping makes them equivalent in size.

Each query contains the address of the host where the reply is to be sent. While originally this was the address of the sender of the query, hosts that indirect the query must change this sender address to match their own, because otherwise these indirected packets could be distinguished (by the receiver) from packets that originate from the host itself, because they have a different return address. The host must keep track of the queries that it indirected so that it can forward a reply to the host where the query originally came from. This statefulness of the routing is probably the biggest scalability issue in GNUnet.

5.3 GNUnet is Malicious.

In GNUnet, the "malicious" behavior described in the section 4.3 is considered to be good. Nodes usually increase k if they receive more traffic than they are willing to handle. Thus, if nodes receive a great deal of traffic, they can improve their performance by reducing the number of packets they indirect. Because the replies are significantly bigger than the queries, this behavior can improve the situation, in particular, for bottlenecks in the network.

It is possible that this behavior could be attacked by flooding a host, A, with traffic from a malicious host, M. Indirected queries originating from M do not count toward m in the formulas given above because the adversary knows that they do not come from A. If A decides that traffic is high and then starts to

[3] In that case, this exchange can not be distinguished from noise for an eavesdropper.

preserve the sender addresses of most queries from non-malicious hosts, m may decrease so far that A can no longer hide its own n queries from being guessable by M.

GNUnet guards against this attack by dropping queries from hosts that are considered malicious. Malicious hosts are defined as all hosts that send excessive amounts of queries (see also [3]).

This adaptive scheme can thereby give the same guarantees as the "always indirect" systems, while being nearly as efficient as the non-anonymous "never indirect" systems:

- If challenged, nodes can always claim that they indirected a query.
- Nodes that are very busy can still refuse to indirect, yielding the same load as for the "never indirect" system.
- For nodes that are not busy, the extra load imposed by the indirection is not a problem.

5.4 Choosing the Next Host

Whenever a GNUnet node receives a query it decides to how many hosts it will forward the query based on its load (CPU, network), the local credit rating of the sender, the priority of the query and some random factor. The number n of hosts that will receive the query could be zero. The n hosts that will receive the query are then chosen from the list of hosts that the node has established connections with using a biased random selection process. The selection process is slightly biased towards hosts where the hash of the hostkey is close to the query using some metric. This is a variant of the algorithm used by Pastry [12, 9].

The query is not sent immediately to the next hosts. Instead, it is put in a buffer that queues data that is to be sent to that host. The buffer is send whenever it is full, a randomized timer goes off, or discarded if the node decides that it is too busy (the protocol does *not* provide reliable delivery).

This behavior does not leak any information to an attacker as it is independent from the original sender (the originator has even an identical chance to receive the forwarded query as everybody else has), it does not reveal anything about the potential source of the reply (the indirecting host does not know that source) and reveals very little about the query: the bias takes 32 of the 160 bits of the hashcode into account. Thus 2^{128} queries would result in the same biased behavior, even under the assumption that the bias would be sufficient to identify the 32 bit.

Replies are indirected back on the path that the query took originally. This does not reveal much about the sender. One attack would be a timing analysis that looks at the time that passes between query and reply. That time could be used to estimate the distance to the sender. In GNUnet, the sender introduces some delay on the reply, making this harder. Furthermore, hosts choose the routes for a query at random, making the timing results hard to reproduce.

5.5 Measuring Anonymity

From the description above it should be clear that the degree of anonymity that GNUnet offers can be configured. If a node injects a query from the user in only 1 of 1000 indirected transactions it will surely be more anonymous than if it injects a query in 1 out of 10 transactions.

Still, the exact degree of anonymity, the probability with with the adversary can say who was the originator, can not be computed in the general case. The reason is that this would require knowledge about how much traffic is controlled by the adversary. As stated in section 2.2, if the adversary controls the traffic of the entire network, it can determine with certainty from where an action originated.

Estimating the number of adversary-controlled hosts in an open network like GNUnet is obviously very difficult. Proving the authenticity of a remote microprocessor [8] and ultimately the trustworthyness of a remote machine is still an open problem.

GNUnet compensates for the impossiblitiy of guaranteing anonymity against very powerful attackers by providing deniability [2]. Even if a powerful adversary can find out who send a message, the adversary may still have no idea what the message was about.

6 Conclusion

In this paper we have described why traditional schemes for anonymity are both inefficient and, often, insufficient if traffic analysis is used. An adaptive indirection scheme was described that is based on the current network load which yields better anonymity guarantees and is more efficient.

References

1. S. M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communication Review*, 19(2):32–48, 1989.
2. K. Bennett, C. Grothoff, T. Horozov, and I. Patrascu. Efficient sharing of encrypted data. In *Proceedings of ASCIP 2002*, 2002.
3. K. Bennett, C. Grothoff, T. Horozov, I. Patrascu, and T. Stef. Gnunet whitepaper, 2001.
4. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
5. I. Clarke. A distributed decentralised information storage and retrieval system, 1999.
6. D.E. Comer and D.L. Stevens. *Internetworking with TCP/IP Vol.II - Design, Implementation & Internals.* Prentice Hall, Englewood Cliffs, NJ, 1991.
7. K. Hickman. The ssl protocol. internet draft rfc, 1995.
8. Rick Kernell. Proving the authenticity of a remote microprocessor, 2002.
9. Y. C. Hu A. Rowstron M. Castro, P. Druschel. Exploiting network proximity in peer-to-peer overlay networks.

10. M. Reed, P. Syverson, and D. Goldschlag. Proxies for anonymous routing, 1995.
11. Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
12. Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems.
13. IETF Secure Shell (secsh) Working Group. Secure shell (secsh) charter., 1999.
14. P F Syverson, D M Goldschlag, and M G Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 4–7 1997.