JANUS: Server Anonymity in the World Wide Web

Andreas Rieke
ISL Internet Sicherheitslösungen GmbH, Germany

Thomas Demuth
FernUniversität Hagen, Germany

About Andreas Rieke

Andreas Rieke received the degrees Dipl.-Ing. (FH) in 1990 and Dipl.-Ing. in 1993 from the College of Bielefeld, Germany and the University of Hagen, Germany, respectively. From 1991 until 1992, he worked in the Laboratory for Applied Mathematics at the College of Bielefeld.
Since 1993, he worked at the University of Hagen or the Forschungsinstitut für Telekommunikation, Dortmund, Germany, as research assistant and finished his Ph. D. thesis "Encryption in ATM systems" in December 1999. After that, he founded the ISL Internet Sicherheitslösungen GmbH which deals with network security and privacy services in the Internet.

Mailing Address: ISL Internet Sicherheitslösungen GmbH, Feithstr. 142, 58097 Hagen, Germany, Phone: ++49(0)2331/3779401, Fax: ++49(0)2331/3779406, Email: andreas.rieke@isl-online.de

About Thomas Demuth

Thomas Demuth is research assistant at the University of Hagen, Germany since 1996; there he is working at the department for communication systems. After studying computer science and attaining his M. Sc. degree in 1993 he worked for EDS, a global company for information technology. Focus of research are security and copyright protection in open networks, especially the World Wide Web.

Mailing Address: FernUniversität Hagen, Feithstr. 142, 58084 Hagen, Germany, Phone: ++49(0)2331/987-4163, Fax: ++49(0)2331/987-397, Email: thomas.demuth@fernuni-hagen.de

Descriptors

JANUS, Anonymity, Privacy, Rewebber, Anonymizer

JANUS: Server Anonymity in the World Wide Web

Abstract

A lot of effort has been undertaken in recent years to hide the content of a message from eavesdroppers. However, often not only the content, but also the address and identity of sender and/or receiver of the message are of interest for attackers. For that reason, several approaches were developed to guarantee anonymity in case of email.

Nowadays, some projects deal with various mechanisms to guarantee client anonymity in the world wide web (WWW), but until today there is no server anonymity service. Based on Chaum's solution for untraceable electronic mail, we introduce a novel solution for server anonymity which enables anonymous publishing in the WWW and present a prototype implementation called JANUS that supports both client and server anonymity.

Introduction

After hiding the content of a message has been a subject of various research projects in the last years, anonymity of communication is becoming a more and more important topic.

Anonymity of communication has been classified for example in (Pfitzmann & Waidner, 1987) in the following way:

- Recipient Anonymity:
  The receiver of a message is kept anonymous. A simple but impractical method to realise recipient anonymity is to send the message to a large number of stations (broadcast or multicast).
- Unlinkability of Sender and Receiver:
  Unlinkability of sender and receiver hides the relation between sender and recipient of a message.
- Sender Anonymity:
  Using sender anonymity, the receiver of a message has no idea who sent the message to him.

Anonymity in case of email has been investigated in several projects. However, the exponential growth of the internet was not caused by the possibility to send electronic mail, but by the world wide web (WWW) which is mainly based on the hypertext transfer protocol.

Since HTTP and related protocols do not use the terms sender and recipient as email does, we introduce the notion of client and server anonymity in this context. Only some projects have analysed anonymity in the world wide web, and most of these have restricted their attention to client anonymity. Some of them also addressed unlinkability of client and server, but server anonymity has not yet been analysed.

In this paper, we introduce server anonymity in the world wide web and present a first prototype implementation called JANUS (justly anonymising numerous URLs systematically) that supports both client and server anonymity.

The rest of this paper is organised as follows:
In section 2, the basic idea of client anonymity is given. A summary of approaches that guarantee client anonymity is included. The next section describes the concept of mixes which was developed by David L. Chaum.
In section 4, we introduce the idea of server anonymity, give examples where the idea is useful and briefly describe how to realise server anonymity. In the next section, a first prototype implementation is described. Section 6 presents some known problems, before section 7 concludes the paper.

Client anonymity

Client anonymity means that in a communication process according to the client server model all information about the client is concealed, i. e. the client is kept anonymous.

There are two ways for a server or an eavesdropper to gain information about the client:
1. In the internet protocol (IP), each packet contains its source and destination address. Thus, eavesdroppers or the server can get knowledge about the client's host in case of a direct connection.
2. The data exchanged between client and server may contain information that reveals the client. In case of HTTP which is a frequently used protocol, the client software (for example, Netscape Communicator or Microsoft Internet Explorer) also may reveal information like the client's email address or the referring page.

In the following subsections, approaches that guarantee client anonymity are described. Each of these has been implemented at least once. Since client and server software is widespread, we disregard approaches that require new user agents or server software.

Proxies

Proxies receive requests from clients and forward these requests to web servers. In the usual meaning, they cache information and thus provide fast retrieval of often used pages and avoid unnecessary traffic.

In contrast to usual proxies, proxies in the sense of this paper anonymise both requests and responses (i. e. links are changed in a way that the contents will also be retrieved anonymously). Elements which endanger anonymity are removed.

Drawbacks of this approach are:

- The user has to trust the proxy.
- Although chaining of requests is possible, it provides no advantages.

The Anonymizer (http://www.anonymizer.com/) was one of the first anonymity services offered in the internet based on the proxy approach.

To demonstrate the importance of anonymous connections, the Anonymizer offers a page where all information about a user is shown. One example is:

> *Your name is probably Andreas Rieke, and you can be reached at rieke@corona.fernuni-hagen.de.*
> *Your connection provider is located in Germany (Federal Republic of).*
> *Your computer is a Unix box running SunOS 5.5.1 sun4u.*
> *Your Internet browser is Netscape.*
> *You are coming from corona.fernuni-hagen.de.*
> *You just visited the Anonymizer Home Page.*

Another, more comprehensive approach, the Lucent Personalized Web Assistant (Bleichenbacher & Gabber & Gibbons & Matias & Mayer, 1998; Gabber & Gibbons & Matias & Mayer, 1997; http://lpwa.com:8000/), was carried out at Bell Labs and Lucent Technologies. Besides anonymity, privacy and account security are also provided.

Crowds

Crowds operates by grouping users into a large and geographically diverse group (crowd) that collectively issues requests on behalf of its members. Instead of directly requesting a document from a web server, a member of a given crowd forwards the request to another,

randomly chosen member of the crowd. That member either submits the request directly to the web server or forwards the request to yet another, randomly chosen member. In the latter case, the next member chooses to submit or forward the request independently using a given probability p_f.

Web servers are unable to learn the true source of a request because it is equally likely to have originated from any member of the crowd. Assuming a number of collaborating (corrupt) members, these might try to find out who was the true source of a request. Since the member that issued the request to the first of the collaborating members might do this on behalf of another member, the true source can not be determined.

Since traffic between the members is encrypted, a local eavesdropper near the sender is not able to learn about neither the content nor the receiver. However, the crowds system has some drawbacks:

- A crowd member may be suspected to have issued a request that was issued automatically on behalf of another user.
- The probability of forwarding a request p_f must be fixed to a suitable value:
  - In case of collaborating users, the first of these can assume that the member that issued the request is the true source of the request. The lower p_f is chosen, the higher is the probability that this assumption is true.
  - Setting p_f too high results in an increasing number of crowd members for a transaction which decreases performance, speed and reliability of the transaction.
- The number of members of the crowd n should be large.
- It is unlikely that all crowd members are trustworthy. Thus, we have to assume malicious members. Because the requests and contents are accessible by the malicious members, they are able to collect sensitive information like passwords or credit card accounts.

The Crowds project (Reiter & Rubin, 1997; http://www.research.att.com/projects/crowds/) was carried out at the AT&T Labs.

Onion Routing

In the onion routing approach, a request (not necessary a HTTP request) is first sent to a client proxy which is operated by the client. The client proxy calculates a route through one or more core proxies.

The client proxy also encrypts the request and the address of the server using the public key of the last core proxy on the route. This is the kernel of the onion.

The layers of the onion are given by the other core proxies. For each of them, content and address are encrypted by the client proxy using the core proxies public key. The encrypted content is then forwarded to the first core proxy which decrypts the message using his private key. After decryption, this core proxy receives address and content for the next core proxy and forwards the request. This is continued until the request is sent to the server.

The onion routing approach has the following drawbacks:

- Since the whole content is en- and decrypted using public key cryptology, this approach is very time consuming.
- In case of a firewall, the approach requires a proxy operating on the firewall.

A more detailed description of the onion routing approach can be found at http://www.onion-router.net/ and in (Syverson & Reed & Goldschlag, 1997; Reed & Syverson & Goldschlag 1998). Although this approach is similar to that we present in the next section, there is a big difference between the two approaches which we will address in section 4.

Mixes

In 1981, David L. Chaum published an idea (Chaum, 1981) which enables untraceable electronic mail, return addresses, and digital signatures. In this approach, the sender of an email computes a route to the receiver which crosses several intermediate nodes called mixes. Each of these mixes provides a public key cryptosystem with the property that

$$d(e(x)) = x = e(d(x)),$$

where $e(x)$ denotes the encryption function and $d(x)$ is the decryption function. In the following we will assume that the public keys of the mixes are available for all involved components in an authenticated way.

Before sending the email, the sender A encrypts the mail using the public encryption function of the receiver. The receiver's address is appended to the encrypted content; after that, the concatenation is encrypted again using the public encryption function of the last mix in the route. In the next step, the address of the last mix is added to the message and both is encrypted using the public encryption function of the previous mix.

This is continued until the first mix of the route is reached. After encrypting the content for that mix, the email is sent there.

Each mix that receives a message first decrypts it using its private decryption function. From the decrypted message, the message for the next hop and its address are obtained and the message is sent there.

As an example, a message M is sent to address A_B using two mixes with address A_1 and A_2 which provide the encryption functions $e_1(x)$ and $e_2(x)$. The receiver's encryption function is $e_B(x)$. Using "," as a symbol for concatenation, the message sent to the first mix A_2 is

$$e_2(e_1(e_B(M), A_B), A_1)$$

If x is simply encrypted using $e(x)$, anyone could test whether $y = x$ by checking whether $e(y) = e(x)$. In order to eliminate this threat, a large string of random bits R is added to x before encrypting. Using this improvement, the message sent to the first mix is

$$e_2(e_1(e_B(M, R_0), A_B, R_1), A_1, R_2)$$

Using this approach, sender anonymity is guaranteed; by reversing it, receiver anonymity is reached.

In order to enable the receiver B of an anonymous email to send a reply to the sender A without getting knowledge of the senders address or identity, the sender has to include his encrypted address in his email. For that reason, the sender computes a route from the receiver back to himself using several mixes as intermediate nodes.

Beginning at the mix nearest to himself, he encrypts his address using the public key of the mix. The encrypted address is encrypted once more with the public key of the next mix. This is continued until each mix has encrypted the address once. Using n mixes, the address now looks like

$$e_1(R_1, e_2(R_2, ...e_{n-1}(R_{n-1}, e_n(R_n, A_A))...))$$

With the function $r_i(x)$ which works with R_i as secret key, B and all mixes encrypt the content of the response since R_i is known only to the specific mix and A. Thus, B sends

$$r_0(M)$$

and A receives

$$r_n(r_{n-1}(\ldots r_2(r_1(r_0(M)))\ldots)).$$

Server Anonymity

The approaches described in section "Client anonymity" only provide client anonymity which means that the identity of the client is not accessible by the server. In the rest of this paper we will concentrate on both client and server anonymity. In this case, neither client nor server know the identity of the other one.

Why server anonymity?

To answer the question where this idea is useful, consider following examples:

1. The review of research papers for a scientific conference is often performed anonymously, that means, both the reviewer and the author of a paper shall not get knowledge of the identity of the other one. For that reason, papers are submitted via a trusted third party without the name of the author and review results are written without revealing the identity of the reviewer.
   However, authors often want to cite their own recent work, but they may not include obvious references in their papers. A solution for this problem is to publish the authors recent papers anonymously in the Internet without revealing any information about their identity.
2. An electronical newspaper might try to publish advertisements of their readers. Since in some cases (for example, employment advertisements) the readers do not want their address or identity to be published, the newspaper assigns a number to those advertisements and acts like a mix because it receives the responses to the advertisements and forwards them to the reader.
   In the electronical version, the user might send the encrypted address of his web server to the newspaper which just publishes this address. In this scheme, not even the newspaper knows the address or identity of the reader.
3. In the Yugoslav civil war the internet was one of the few possibilities for civil right groups to communicate with the rest of the world. They were able to report about the injustice of their government by email and news, but not in the WWW because of the danger of backtracking. In this case and in other totalitarian countries server anonymity is able to guarantee the anonymity of speech at least in the internet which avoids persecution where freedom of speech is absent.

How to realise client and server anonymity

Based on Chaums idea, there are several points to pay attention to in order to provide server anonymity in the world wide web. One of the first is the transfer protocol. Although HTTP is the best known protocol nowadays, other, related protocols like FTP or Gopher can be treated in the same way.

The transfer protocol carries not only metainformation about the content, but also address information (for example, the URI (uniform resource identifier) of the page to get) or other information about client or server (for example, the browser's or server's software version) which might reveal information about server or client.

The next point to pay attention to is the transferred content. From the viewpoint of a mix, not only content is transferred, but also address information since, for example, content of type HTML (hypertext markup language) usually contains links to other pages. In contrast to the onion routing approach, it is important to detect which part of a message is real content and which part is address information since it has to be processed in an appropriate way.

Misuse

We define misuse as the usage of server anonymity in a way which offends against german, european or international law, affects the operation of other servers or parts of the internet or violates common moral standards.

Unfortunately, the interest in using server anonymity in an illegal way might be quite high. Assume a dealer who deals with illegal content, for example pornographic images. In the past, his main problem was that his identity could be revealed since his server address was known. Nowadays, using server anonymity, he is able to deal with his contents without enabling anyone to find out the servers address.

JANUS: The prototype implementation

Our prototype implementation is called JANUS (justly anonymising numerous URLs systematically) according to Janus, the Roman god of doorways who had two faces.

In the prototype implementation, we provide access only to public pages, do not encrypt the transferred content, and restrict our attention to handling address information. Although this enables eavesdroppers to reveal information about the content, it has the following advantages:

- We are able to detect and stop misuse. If the content were encrypted, we would not even be able to distinguish HTML pages from graphics.
- The client and server agent functionality may be integrated into the mixes allowing only a single type of agent which is shown in fig. 1.
- Due to less en- and decryption, this solution provides higher speed.

Our prototype is also vulnerable to a timing attack, since we do not insert delays.

Intruder Model

Due to the missing encryption of transferred content, the intruder model of our prototype implementation is not the same as in Chaum's scenario.

In the original mix concept an intruder can tap all lines and intercept all but one mix without being able to find the relation of the sender and receiver of a message.

Using client and server anonymity in the JANUS prototype implementation, we define two areas of trust: the client and the server area. Each of these areas contains the client or server, respectively, the line from the client/server to the next JANUS server and that JANUS server itself.

The relation of client and server is obvious to everybody that has access to both areas.
For example, the client gets knowledge of the server's address if he taps the line from the server to the JANUS server connected to the server.

For this reason, it makes no sense to use more than two JANUS servers since additional ones do not improve the security of the system.

Transfer Protocols

JANUS at present provides HTTP/1.0 (Berners-Lee & Fielding & Frystyk, 1996) and HTTPS (with 128 bit encryption) as transfer protocol between JANUS and clients, and supports HTTP, FTP and Gopher to access servers. Since HTTP is the most interesting protocol, we restrict our attention to HTTP in the following.

Besides the request or status line, HTTP consists of several header fields which contain additional information. In principle, we decided to let pass all HTTP/1.0 header fields and to remove all other fields. Since cookies are not part of HTTP/1.0, transmission of cookies is not allowed.

Transferred content

The content is of interest only if it might contain address information, since otherwise it passes JANUS unchanged. In the following, we restrict our attention to HTML content (Raggett, 1997).

To avoid the risk of loosing anonymity, some tags and attributes are removed in any case. For example, Java and Javascript elements are removed in this way.

Attributes that contain address information are expanded to absolute addresses and then encrypted.

Encryption

For encrypting and decrypting address information, we make use of the RSA public key cryptosystem (Rivest & Shamir & Adleman, 1978) with a 768 bit key. Keys are generated according to (Gordon, 1984); in order to speed up encryption, we use 65537 as public key e.

In contrast to (Chaum, 1981), we do not need to encrypt random bits since we do not encrypt the content and comparisons of encrypted address information is allowed in our case; otherwise, an opponent might compare the contents he received through JANUS.

In (Pfitzmann & Pfitzmann, 1990), an attack to the direct RSA-implementation of mixes based on the multiplicative structure of RSA is given. In our implementation, the attack does not lead to any success since we provide enough redundancy in the address information to encrypt.

Known problems

Prevention from misuse

In order to prevent misuse, we

- could switch off server anonymity or publish our secret key in order to use JANUS as a scientific construct without practical relevance.
  We are not willing to do that since we think that we stop lots of legal usage otherwise. But if misuse outweighs legal use, we might decide to change keys frequently in order to prevent the publication of the encrypted address or to provide the decryption function.
- could publish the encrypted and decrypted address if misuse occurred. Since the operators of these servers can then no longer operate anonymously, they have no more advantages using JANUS.
  The problem in this way is that some people are interested particularly in lists containing address information with illegal content.
- will decrypt address information on behalf of a governmental office and can even provide them access to our decryption function without revealing the secret key.
- will maintain unpublished lists with address information for illegal contents. For each request which is carried out using JANUS, JANUS will check whether the address is contained in the list and if so JANUS will deny access to the content.

Additionally, we log each access in the same way as usual web servers do.

Safe anonymity

In order to provide some pages using server anonymity in the internet, pay attention to the following rules:

- Use as little address information as possible. Address information is recognised as address information only in several HTML tags (for example, in the A-tags HREF attribute), but not in the normal text of an HTML page.
- Restrict access from search engines to your pages. A possible attack to find out your address is to perform a search using keywords from your pages.
- Avoid unusual constructions containing address information that might not be recognised as address information by a JANUS server.

- Test the system by retrieving the pages through single JANUS servers and have a look at the HTML source.
- In the future, try to use different trustworthy JANUS servers.
- Don't publish illegal content since the server operators might be forced to decrypt your address and make it known to governmental authorities.

Speed

Unfortunately, there are a lot of factors that decrease speed:

- Instead of transferring the content once in case of a direct connection between client and server, it has to be transferred once more for each JANUS server involved.
- Since the client provides the encrypted address for each link, decryption has to take place once for each JANUS server.
- In case of content that may provide address information (for example HTML pages which may contain links to other pages), JANUS has to
  - parse the content in order to separate real content and address information. In contrast to the Netscape and Microsoft web browsers, our parser starts parsing not before the whole content has been transferred.
  - encrypt each address information in order to provide server anonymity.

Conclusions

After anonymity in case of email has been analysed and realised in several projects in recent years, some projects have started to work on anonymity in the world wide web. Most of these concentrate on client anonymity, and some also address unlinkability of sender and receiver. In this paper, we present a novel approach for server anonymity and offer a corresponding service in the internet.

Since the interest of using server anonymity in an illegal way might be quite high, we decided to restrict our attention to address encryption leaving the transferred content unchanged. For that reason, we are able to detect and stop misuse.

The JANUS prototype implementation is based on HTTP, HTTPS, FTP and Gopher as transfer protocols. An HTML parser is able to distinguish content from address information. For encryption, we use the RSA public key cryptosystem.

After the JANUS prototype started operation in November 1997, it receives up to 1 000 000 requests per day at the time of writing this paper.

Depending on further experience, we plan to build up a network of JANUS servers in several countries in the future. These will be extended step by step (content encryption, message length variation, message delay, ...) in order to realise real mixes according to Chaum (1981).

References

Bleichenbacher, Daniel & Gabber, Eran & Gibbons, Phillip B. & Matias, Yossi & Mayer, Alain (May 1998). On Secure and Pseudonymous Client-Relationships with Multiple Servers. Technical report, Bell Labs - Lucent Technologies, Murray Hill, NJ.

Berners-Lee, T. & Fielding, R. & Frystyk, H. (May 1996). Hypertext Transfer Protocol - HTTP/1.0. RFC 1945.

Chaum, David L. (February 1981). Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM, 24(2):84-88.

Gabber, Eran & Gibbons, Phillip B. & Matias, Yossi & Mayer, Alain (1997). How to Make Personalized Web Browsing Simple, Secure, and Anonymous. In Rafael Hirschfeld, editor, Proceedings of Financial Cryptography '97, number 1318 in LNCS, pages 17-31. Springer, Berlin.

Gordon, J. (June 1984). Strong RSA Keys. Electronics Letters, 20(12): 514-516.

Pfitzmann, Birgit & Pfitzmann, Andreas (1990). How to Break the Direct RSA-Implementation of MIXes. In Jean-Jacques Quisquater, editor, Advances in Cryptology: EUROCRYPT '89, number 434 in LNCS, pages 373-381. Springer, Berlin.

Pfitzmann, Andreas & Waidner, Michael (1987). Networks without User Observability. Computers & Security, 6: 158-166.

Raggett, Dave (January 1997). HTML 3.2 Reference Specification. W3C Recommendation, W3C.

Reiter, Michael K. & Rubin, Aviel D. (August 1997). Crowds: Anonymity for Web Transactions. DIMACS Technical Report 97-15, AT & T Labs, Murray Hill, NJ.

Rivest, R. L. & Shamir, A. & Adleman, L. (February 1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21(2): 120-126.

Reed, Michael G. & Syverson, Paul F. & Goldschlag, David M (May 1998). Anonymous Connections and Onion Routing. IEEE Journal on Selected Areas in Communications, 16(4): 482-494.

Syverson, Paul F. & Reed, Michael G. & Goldschlag, David M. (1997). Private Web Browsing. Journal of Computer Security Special Issue on Web Security, 5(3): 237-248.

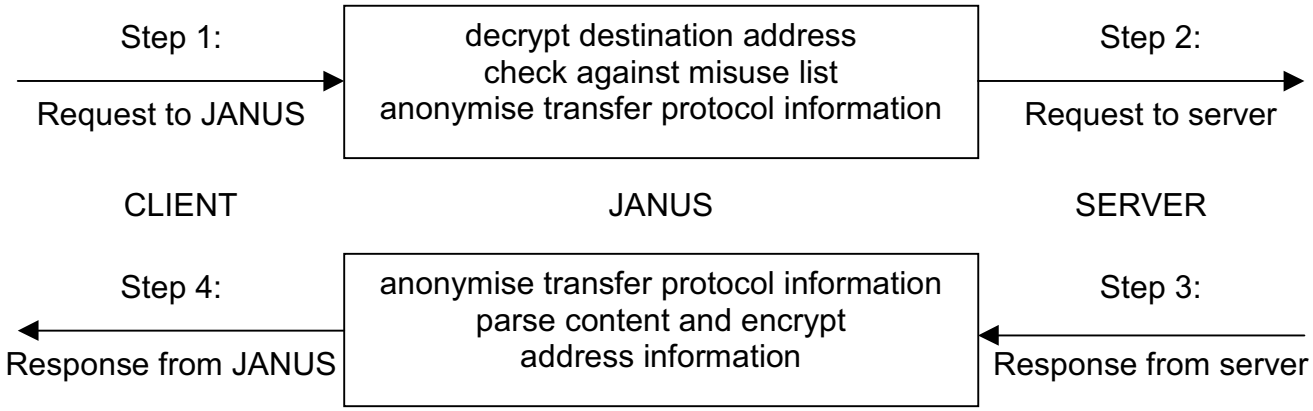

Figure 1: GETTing information via a single JANUS server