SECURE AND ANONYMOUS MULTICAST FRAMEWORK

Nathalie Weiler, Bernhard Plattner
Computer Engineering and Networks Laboratory (TIK),
Swiss Federal Institute of Technology ETH Zürich, Switzerland
{weiler, plattner}@tik.ee.ethz.ch

Abstract

The rapid increase in Internet users triggered a number of new Internet services and applications such as online shopping, video conferencing, Internet games or distance education. A larger part of those ones requires multicast support for efficient data distribution. A number of secure group communication protocols have been published recently, but the preservation of privacy of the single group member is still an unsolved problem. This paper presents a novel approach to secure and anonymous group communication. First, we propose a solution for anonymity in a local environment based on state-of-the art approaches such as pseudonym servers and anonymizers combined with encryption techniques on different protocol levels in order to guarantee an anonymous way of communication between end-users. Then, we introduce the secure and anonymous multicast (SAM) framework and we show how it can be used as a configurable, scalable architecture in combination with local anonymity.

Keywords: Scalable end-to-end anonymous communication, composable privacy, anonymous multicast.

1. INTRODUCTION

The fast growing proportion of the population using the Internet as a means of information exchange comes along with an increase of the opportunities to violate each person's privacy by gathering this information. Typically, cryptographic techniques are used to hide the content of messages by means of encrypting this data. However, thereby the identities of the message's sender and recipient(s) are not concealed. Anonymous bidding schemes in the Internet, e.g., could be implemented more conveniently if the identities of the bidders were hidden. Such schemes could be used in the auction of UMTS (Universal Mobile Telecommunications System) licenses currently taking place in several

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35413-2_36

R. Steinmetz et al. (eds.), *Communications and Multimedia Security Issues of the New Century*

European countries or in any other kind of distributed bidding scheme. Insider information gathering could be prevented, too: If the executives directors of two financial institutions had concealed their identities in the e-mails sent back and forth between them, a system administrator's attention would never have been attracted by this increased amount and the news about an upcoming fusion would never have spread.

The realization that a solution is needed for providing privacy and anonymity in the Internet is neither new [Cha81, Pfi90] nor unexplored (see Section 2). However, the anonymity aspect has seldom been regarded in the context of group communication. The emergence of group applications such as distance education or video conferencing requires for privacy enhancing measure in the MBone – the multicast capable overlay network of the Internet. Levine and Shields [LS00] presented a first approach to achieve receiver anonymity by sending IP packets to a multicast group and thereby preventing disclosure of the recipient. Our approach takes one step beyond this approach. The contribution of this paper are two-folded: First, we present an approach to remain anonymous in a local environment. Then, we extend this approach to a protocol for *secure and anonymous multicast* communication (SAM), i.e. a group communication protocol that allows authorized users to communicate with other group members without being identifiable by outsiders and/or insiders.

The outline of this paper is as follows: In Section 2, we start by giving an overview over previous approaches used in unicast scenarios. Then, our design and implementation of a secure anonymous local infrastructure are detailed in Section 3. Section 4 introduces our novel approach to secure and anonymous multicast communication. Finally, we conclude with a short assessment and an outlook of further work in Section 5.

2. RELATED WORK

We first introduce the idea of Chaumian Mixes which is the basis for many approaches. Then, we produce an overview of the techniques used for e-mail anonymity. We continue with anonymity for the Web and the few network related anonymity usages. A detailed analysis of the presented related work can be found in [WP00].

Chaumian Mixes

Chaum's work is the first one to address the problem of hiding the communication pattern in the network itself. According to [Cha81], a sender packs his message M, figuratively spoken, in n envelopes – each one addressed to one of n chosen mixes so that each mix only knows the next hop information.

The purpose of the mixes – placed as intermediate nodes between sender and receiver nodes – is to complicate traffic analysis. However, the price to pay is

not negligible: (a) Each packet's transmission time is significantly increased by the delay introduced by the mixes[1]. (b) Each mix must be able to perform asymmetric de- and encryption. The encryption used must be probabilistic to prevent the observing adverse from the following traffic analysis attack: He takes the outgoing messages and re-encrypts them with the public key of the mix. By comparing the results to the encrypted input messages, he can trace the traffic. As a prophylactic measure, each ciphertext is longer by at least the amount of random bits used than the input message. Since several mixes should be used, the ratio of information and junk bits increases. Nonetheless, the mixes are the basis for many anonymizing e-mail systems: `anon.penet.fi`, cypherpunk remailers, mixmasters, etc.

Mail based Approaches

Mail based approaches are generally classified into three different types:

(1) A **Type 0 remailer**, the simplest system, strips off headers and forwards the remaining message. Examples are `anon.penet.fi` (Not operational anymore) or `www.mailanon.com`.
(2) The class of the **Type 1 remailers** encompasses all remailers that use any variant of layered encryption such as cypherpunk systems.
(3) Mixmasters or **Type 2 remailers** are more resistant against spamming and traffic analysis attacks.

Web based Approaches

Since the WWW is probably the most frequently used application on the Internet, the demand for anonymous Web browsing is increasing:

`www.anonymizer.com` [Ano] – a service of Anonymizer, Inc. – provides the same service as a type 0 remailer for HTTP traffic. Functionally, the Anonymizer is a web proxy that removes all personal and identifying information from the HTTP request and then forwards it to the destination web servers. The replys are relayed in a similar manner.

The **Lucent Personal Web Assistant** [GGK+99] offers a pseudonym service, i.e. a pair of anonymized username-password, for each requested web server. The idea behind is to prevent several web sites form coordinating their log based informations (IP address, e.g.) about a certain user.

A **crowd** [RR98] is a group of users who collectively perform HTTP requests in order to enable each member to browse the WWW anonymously. From the perspective of the Web server, each member of the crowd is equally likely to have issued the request. Anonymous web traffic performs a secret random walk through the network. Therefore, each member of the crowd runs a *jondo* that

[1] Each mix must wait until the amount of message is sufficiently large enough to mix.

forwards both other crowds member's and the own user's requests. These secret paths are set up in an initial step. After this step, each member uses exactly one same secret path for all subsequent traffic originating at his place. As jondos cannot tell if the request has been issued by its own user or forwarded by another jondo, users remain anonymous. The *advantages* of a system like crowds are that (a) it requires only little low cost – symmetric – encryption and decryption. (b) it can handle network failures efficiently as every jondo can determine the endpoint of the request and reroute the traffic if necessary[2]. (c) the membership in the crowds group is dynamic. The major *disadvantage* of crowds is its vulnerability to combined timing and collusion attacks: HTTP requests frequently tend to appear in bursts, i.e. the initial request triggers several more requests for additional HTTP objects. An insider, incorporating a crowds member, knowing the other member's processing speeds can reveal the true path position of the original request from analysis of the intervals and delays between requests. Furthermore, crowds does not protect from forwarding tracker attacks.

The previous approaches are concerned with the browsing aspect of the WWW. **TAZ Servers** and **the Rewebber Network** [GW98] cope with the publishing aspect of the WWW. In some cases, anonymous publishing on the WWW is desired[3]. Technically, the Rewebber network consists of a cascade of type 1 remailers for HTTP traffic.

JANUS [DR99] has some similarities with the above mentioned rewebber network: Both use layered encryption in a cascade of untrusted HTTP proxies to achieve server anonymity. The main difference is that JANUS parses the HTML documents retrieved and carefully replaces any embedded URLs that might reveal the anonymous original publisher.

Network based Approaches

There exist very few approaches coping with anonymity in Internet connections in general. Most approaches are concerned about a certain application, typically e-mail or WWW.

The **Onion routing** network [RSG98] consists of a number of routers that use forwarding through a cascade of untrusted third parties. The routers maintain a set of encrypted TCP connections to each others. Figure 1 illustrates how the client Andrea contacts the anonymous network through the application proxy of the onion router No. 2. Andrea's onion proxy prepares the onion – i.e. a layered forwarding address structure containing for each one of the used onion routers the next hop information and key seed material (for the generation of the symmetric keys that will be employed by the onion router during the actual

[2]This is the reason why crowds provides only sender anonymity and cannot be extended to receiver anonymity without a severe loss of attractiveness.

[3]Consider for example the potential of such a system for conferences requiring anonymous submissions.

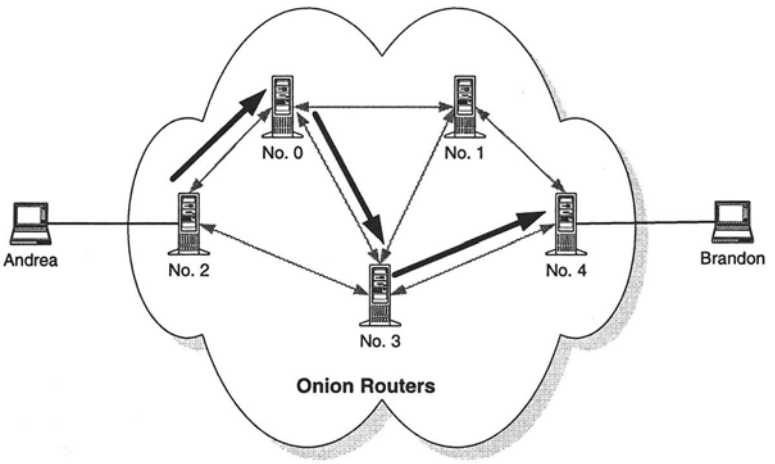

Figure 1 Operation of an Onion Router Network.

routing of the data) – and sends it to the application proxy. The prepared onion defines the path, marked with black arrows, through the onion routers.

Besides being transmitted in uniformly sized blocks, the data is mixed, i.e. collected and reordered randomly. However, the experimental prototype shows unfortunate correlations between several data sources. So, despite of mixing, the chances of a successful traffic analysis attack are still considerable. A replay attack can be tackled with nonces in a successful manner. A flooding attack is the promising approach in case of long lasting connections.

PipeNet [Dai] uses a similar approach to Onion Routing. Additionally to the already introduced features of Onion Routing, the PipeNet switches are more resistant against insider attacks.

The **Freedom Network** [GS99] is an anonymizing overlay network running on IP. Its similarities to Onion Routing and PipeNet are strong. However, the Freedom Network suffers from a far worse design flaw: Active attackers incorporating two Anonymous Internet Proxies (AIP) can trace everyone who uses them as first and last AIP.

3. SNAP – SECURE NONLOCAL ANONYMIZING PROTOCOL

The main motivation for our new approach was the lack in the literature of a simple protocol allowing for anonymous end-to-end communication. Almost all approaches require significant changes in the infrastructure while still being vulnerable to simple traffic analysis attacks. So, we required from our system

that it (a) is simple to use, (b) does not require an expensive infrastructure, (c) supplies a solution for the needs of an average user, and (d) provides, if desired by the user, local anonymity.

We believe that the ensuring of privacy in the local network is one of the most desired features of users. No employee wants system administrators sniffing through e-mails. He simply wants to send and receive e-mail or browse Web pages without anybody in his local environment notifying whom he communicates what with. Similar reasoning holds for the home user with regard to his service providers. On the other hand, our approach should not prohibit the usage of any kind of personalized services.

Design of *SNAP*

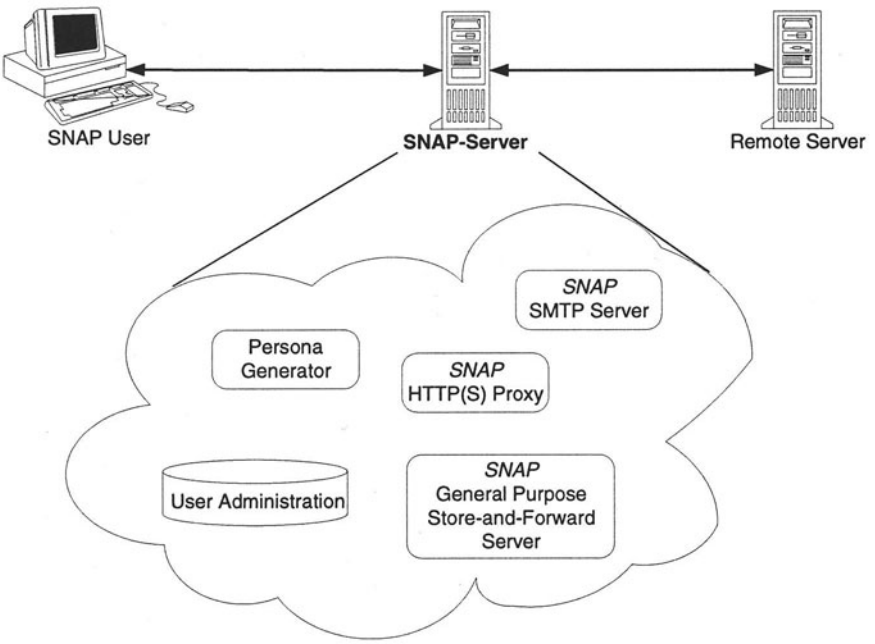

Figure 2 Components in the *SNAP* Architecture.

The major players in *SNAP* (Secure Nonlocal Anonymizing Protocol) are depicted in Figure 2:

(1) The ***SNAP* HTTP(S) proxy** is a store-and-forward HTTP to HTTPS proxy.
(2) The ***SNAP* SMTP server** is a secure remailer handling incoming and outgoing mail traffic.

(3) The ***SNAP* general purpose store-and-forward server** is responsible for all remaining services in the Internet.
(4) The **Persona generator** provides a kind of pseudonyms to the other server components.
(5) The **User administration** handles the profiles of the *SNAP* users.

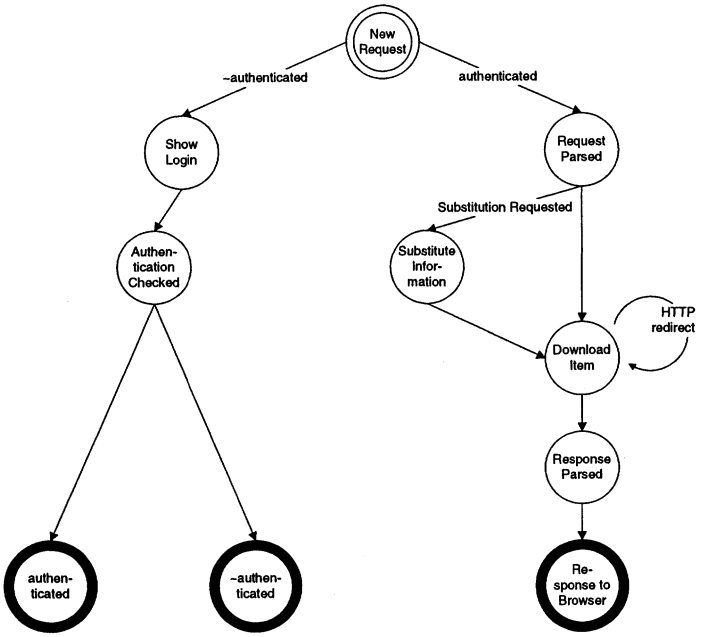

Figure 3 Handling of a new request by the *SNAP* HTTP(S) Proxy.

***SNAP* HTTP(S) Proxy.** *SNAP* requires that every communication using HTTP between user and itself is encrypted. Therefore, the Web browser must request the desired page via HTTPS (using SSL[4]) from *SNAP*. Upon this request, the latter opens an HTTP[5] connection to the target server if the user has already logged in into the system. Otherwise, the user must authenticate himself to *SNAP* and after successful authentication, the proxy requests the objects. The *SNAP* HTTP(S) proxy requests the web object in its name. Then it replaces all links in this object with equivalent links redirected through *SNAP*. Finally, this changed object is sent to the web browser of the user. No additional require-

[4]SSL: Secure Socket Layer.
[5]HTTPS is used if supported by the target web server.

ments are needed either at the user side or at the WWW servers side since any modern browser is equipped with SSL.

As shown in Figure 3, the *SNAP* HTTP(S) proxy asks the user administration component for the login information of the user. If this information is successfully retrieved from the *SNAP* database maintained by the user administration and the persona generator, it is filled in if the GET or POST form contains a substitution order for a pseudonym. In the case that this information does not exist, it is the user's first visit to this hostname and therefore, there exists no pseudonym in the record database. The user administration calls the persona generator for a new pseudonym record consisting of an username, a password and an e-mail alias. The new e-mail alias is simultaneously generated on the *SNAP* SMTP server.

Now the desired object is downloaded from the target web server. The substituted GET or POST parameters are forwarded with the request. The header of the object indicates the type of the object: whether it is a binary file, the object is forwarded through HTTPS to the user's browser. If it as a HTML document, some changes need to be made. Basicly, the Response Parser must filter the javascript , resolve the document base, expand links and rewrite the forms. Then the altered HTML document is transmitted to the browser at the user's side using HTTPS.

***SNAP* SMTP Server.** The *SNAP* SMTP Server is a secure remailer handling incoming and outgoing e-mail. Figure 4 shows the sequence of activities that the *SNAP* SMTP Server must perform.

(1) The local *SNAP* POP Server accepts an E-Mail either for or from a *SNAP* user.
(2) This E-Mail is picked up by the newly created instance of the dispatcher.
(3) The dispatcher's first task is to store the e-mail and all its attachment in a temporary storage.
(4) Depending on the direction of the e-mail, the dispatcher has to start either (a) the encryption engine (in case of an outgoing e-mail that is requested to be encrypted) or (b) the decryption engine (in case of an incoming e-mail that is encrypted.).
(5) After successful completion encryption or decryption, the parser processes the message.
    (a) In case of an incoming e-mail, the remailer information is introduced in the mail header (The sender is set to the remailer's address) and delivered to the addressee.
    (b) In case of an outgoing e-mail, the parser extracts the remailing information supplied by the sender from the body of the e-mail. The mail header is substituted with this information. The resulting e-mail is sent and any temporary files are cleaned up.

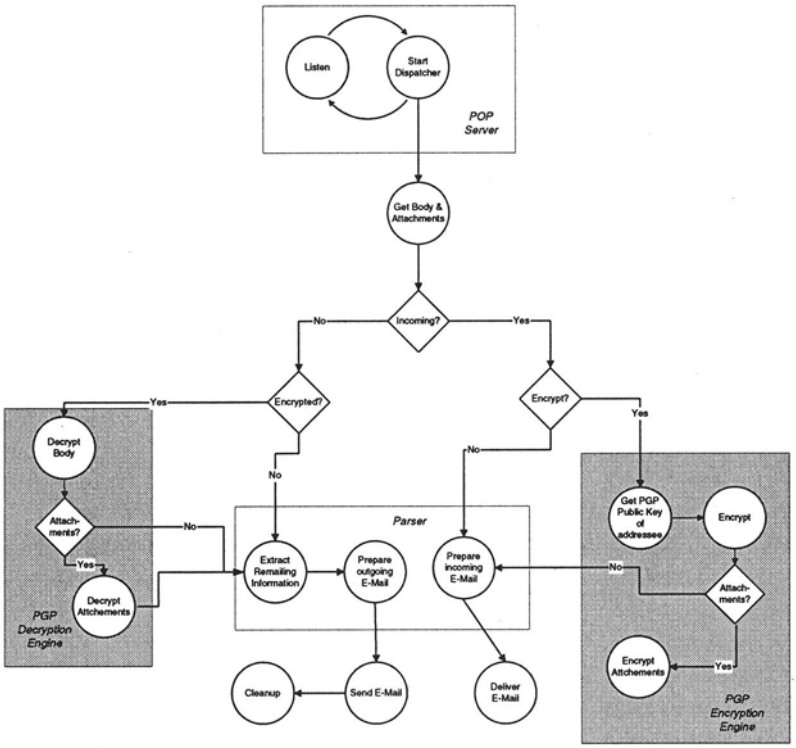

Figure 4 Flow Chart of the Remailing process.

Note that plaintext e-mail is possible. In this case, the encryption/decryption step is skipped as depicted in Figure 4. In this case, the sender or receiver of the e-mail remains no longer locally anonymous, i.e. any observer in the local environment can read the message's body. Furthermore, other weaker encryption mechanisms than Pretty-Good-Privacy [CDFT98] may be integrated into the *SNAP* SMTP Server if public key encryption proves to be a bottleneck of a specific server.

***SNAP* General Purpose Store-and-Forward Server.** The first two components of *SNAP* are concerned about the web and mail traffic. This set is completed with a third component targeted at the remaining traffic. The reasons for separating SMTP and HTTP streams from data of other protocols is two-folded: (1) Secured methods for both e-mail and web objects transmission, e.g. PGP, S/MIME [Ram99], SSL [DA99] exist and are typically already integrated or easy to link to current mail- and webservers. So, *SNAP* may profit form this work already done for providing anonymous SMTP and HTTP

services. (2) Both SMTP and HTTP traffic range under the top five TCP application [Mkc00]. So, separate handling of those TCP connections disburdens the general purpose *SNAP* server.

The *SNAP* general purpose store-and-forward server uses a similar approach than the other two components already introduced: (1) IPSEC [KA98] is the security mechanism used in the Internet Protocol. We use this quasi built-in mechanism to transmit confidential information to the *SNAP* server. (2) The addressing information of the target server is hidden in the payload by encrypting this payload by means of IPSEC. At the client side, the client application must be aware of the anonymous connection, because it must use a modified connection manager. The latter one is responsible for putting the target server addressing information in the payload of the packet and addressing the result to the *SNAP* server.

Persona Generator. The persona generator is responsible for building pseudonyms. It uses a randomizer for providing ten to twenty character long pseudonyms. Duplicates are rejected. The randomizer used is a variant of the Automated Password Generator [AGP].

User Administration. A single working unit of *SNAP* can only be used by registered users[6]. At registration time the user must chose a loginname and a password, and reveal one personal e-mail address. Optionally, he may deposit his PGP public key in order to have it integrated in the *SNAP* PGP public keyring and signed with the corresponding private key of *SNAP*. This information is written on a secure disk. Before each session, a user must authenticate himself.

4. SECURE ANONYMOUS MULTICAST COMMUNICATION

A typical group in a multicast scenario consists of several participants P_i which can be classified into two categories according to their role in the data distribution process: senders and receivers. Each participant may play both roles – e.g. in a distributed game, each player is fed some information by the game server and the other players, but he also provides his moves to the group.

Our novel approach to anonymous multicasting introduces a new participant in the multicast group: the secure and anonymous multicast (SAM) server. Figure 5 depicts an example of two SAM servers A and B joining the multicast group M consisting of the participants P_1, P_2, P_6 and P_7. Through the membership of the SAM server A, P_3, P_4 and P_5 participate anonymously in the group. Similarly P_8 and P_9 join the group M together with the SAM server B.

[6]This design decision was motivated by the wish to reduce additional spam e-mail to a minimum.

All multicast traffic divulged in the group M is transmitted in a secure way by the SAM servers to their respective anonymous subscribers.

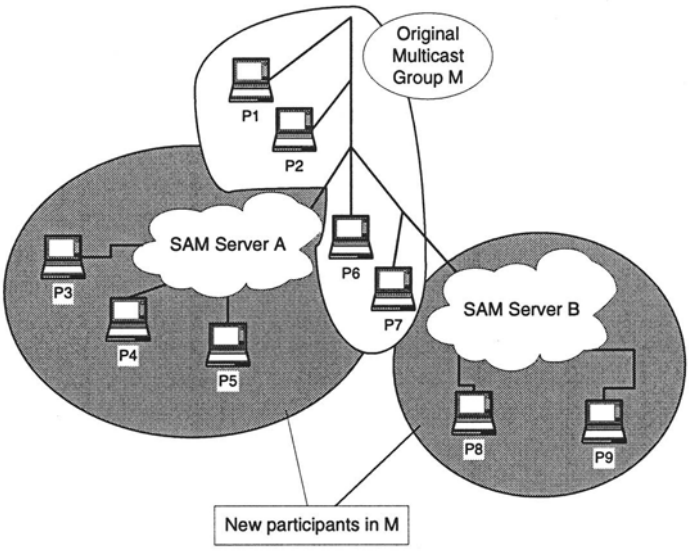

Figure 5 Illustration of the SAM Approach.

The SAM servers joining on behalf of the users who wish to remain unknown to both the group and any outsiders are treated as normal members of the group and play their expected role in the chosen group key management technique. However, the true members, i.e. the anonymized group members receive the secure group communication through the SAM server . Therefore, we use the SNAP architecture described in Section 3. Thereby, we inherit the user administration and persona generator service on one hand and, most importantly, the users are protected from any malicious observer in their local environment due to the encrypted transmission between SAM server and user.

The SAM server, until now referred to as one server per local environment, may consist of a network of SAM servers for *scalability* reasons such reducing the trust required. For further details of the SAM approach, the interested reader is refered to [WP00].

5. CONCLUSIONS

In summary, SNAP is a simple anonymizing protocol that prevents observers in the local environment of the user from learning any information about the

traffic transmitted between SAM server and user[7]. We refer to this property as *local anonymity*. The main building blocks assuring this local anonymity are simple and inexpensive to use as shown in Section 3 for anonymization of unicast traffic.

We extended the SNAP infrastructure to a multicast environment. The resulting SAM framework provides an environment for anonymous group communication build on top of state-of-the-art technology. The exact composition of the framework is *configurable* by the application, e.g. the application decides on the access mechanisms or if encryption algorithms are mandatory. The usage of network of SAM servers for *scalability* reasons reduces the trust required in each of the individual servers.

Further research on this topic will encompass the analysis of the behavior of different applications in the SAM framework with respect to selected authentication methods and group management techniques and the quantitative evaluation of the implementation concerning performance and usability.

Acknowledgments

The authors would like to thank Andri Krämer who implemented parts of the first SNAP prototype. Part of this work was funded by the Swiss National Science Foundation (SPP-ICS).

References

[AGP] Automated Password Generator (AGP). `http://www.itl.nist.gov/fipspubs/fip181.htm`.

[Ano] Anonymizer. The Anonymizer. `http://www.anonymizer.com`.

[CDFT98] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP message format. Internet RFC 2440, November 1998.

[Cha81] David L. Chaum. Untraceable electronic mail, return adresses, and digital pseudonyms. *Communications of the ACM*, 24(2), February 1981.

[DA99] T. Dierks and C. Allen. The TLS Protocol – Version 1.0. RFC 2246, January 1999.

[Dai] Wei Dai. Pipenet 1.1. `http://www.eskimo.com/~weidai/pipenet.txt`.

[DR99] Thomas Demuth and Andreas Rieke. Securing the Anonymity of Content Providers in the World Wide Web. In *Proceedings of SPIE'99*, volume 3657, pages 494–502, San José, CA, USA, January 1999.

[GGK+99] Eran Gabber, Phillip B. Gibbons, David M. Kristol, Yossi Matias, and Alain Mayer. On secure and pseudonymous client-relationships with multiple servers. *ACM Transactions on Information and System Security (TISSEC)*, November 1999.

[7]This property holds as long as the private information used in the decryption process (e.g. the private key and the pass phrase of the user if PGP is used) is not accessible by the observer.

[GS99] Ian Goldberg and Adam Shostack. Freedom Network 1.0 Architecture and Protocols. White Paper, `http://www.freedom.net/info/freedompapers/index.html`, November 1999.

[GW98] Ian Goldberg and David Wagner. TAZ Servers and the Rewebber Network: Enabling Anonymous Publishing on the World Wide Web. *First Monday*, 3(4), April 1998.

[KA98] S. Kent and R. Atkinson. Security architecture for the Internet Protocol. RFC 2401, November 1998.

[LS00] Brian Neil Levine and Clay Shields. A Protocol for Anonymous Communication over the Internet. In *Proceedings of ACM Conference on Computer and Communication Security CCS'00*, 2000.

[Mkc00] Sean McCreary and kc claffy. Trends in Wide Area IP Traffic Patterns: A View from Ames Internet Exchange. `http://www.caida.org/outreach/papers/AIX0005/`, September 2000.

[Pfi90] Andreas Pfitzmann. *Dienstintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz.* PhD thesis, Universität Karlsruhe, Deutschland, Informatik-Fachberichte 234, Springer Verlag, 1990.

[Ram99] B. Ramsdell. S/MIME Version 3 message specification. RFC 2633, June 1999.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), November 1998.

[RSG98] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *Journal on Selected Areas in Communications*, 16(4), May 1998.

[WP00] Nathalie Weiler and Bernhard Plattner. Secure anonymous protocols for local and multicast environments. Technical Report 73, TIK, ETH Zürich, Switzerland, October 2000.