Secure Anonymous Group Infrastructure for Common and Future Internet Applications

Nathalie Weiler
Computer Engineering and Networks Laboratory,
Swiss Federal Institute of Technology, Zürich (ETHZ), Switzerland
weiler@tik.ee.ethz.ch

Abstract

Secure group communication protocols, in particular multi-party key agreement and update algorithms, help promote traditional and new Internet multi-party applications such as video conferencing or distance education. We propose a framework for marrying such approaches with access management mechanisms and applications in real environments. Furthermore, we extend this framework with anonymisation techniques for the sake of the individual's privacy. Our solution combines traditional unicast based approaches for privacy with authenticated and encrypted group communication. Thereby, we are able to build closed groups in which the members are not disclosed to outsiders. The introduced secure and anonymous multicast (SAM) framework can be employed as a scalable, configurable architecture for pseudonym based group communication between qualified entities.

Keywords: scalable anonymity, configurable end-to-end anonymous communication, pseudonymous multicast, anonymous group communication.

1 Introduction

The success of the MBone – the multicast capable overlay network of the Internet – and similar mass communication networks will be determined by their ability both to preserve the privacy of their users and to secure the network infrastructure of their operators. IP multicast does not support closed user groups. In other words everyone can listen to all traffic of a specified multicast group as any host may join or leave a multicast group by sending IGMP [1] messages to their local router. Furthermore, multicast accentuates certain security threats, esp. active attacks such as denial-of-service ones, malicious replays or masquerading, because of missing access control mechanisms [2]. A typical solution for secure multicast requires a many-to-many agreement on the encryption key. Different mechanisms to solve the problems of key establishment and update in such groups have been proposed in the recent past. The communication and computational costs of well-known approaches are shown in [3].

This paper goes one step further and introduces an infrastructure for secure and yet anonymous group communication – allowing for a way of communicating in closed groups without disclosing its privacy. A **secure anonymous group** has an access control mechanism, distributes data traffic encrypted for its members only, and hides all or part of its member's identities to outsiders. In other words, we we understand by secure anonymous multicast (1) that only users who fulfil certain conditions are allowed to join the closed group, (2) that a non-member of the group cannot understand the data distributed, and (3) that the identity of a member may not be disclosed to outsiders of the group. Additionally, the member may hide its identity to other group members, if the policy allows for. A member of a secure and anonymous group presents upon his join in the group the necessary proof that he is entitled to do so. After the join, the member stays anonymous in the group so that no outsider can either identify the sender or the receiver or decrypt the data sent.

Internet based teaching is one of many future applications profiting from such a secure anonymous group infrastructure. Lectures could be followed anonymously, yet the students could be charged by administration of the teaching academy. The examinations could be semi-blinded so that the student's identity is hidden from the examiner allowing for a totally fair qualification, but still the examinee cannot impersonate someone else. Note also that this situation is almost impossible to achieve in traditional non-Internet based examinations. Other application fields include anonymous review processes or virtual conferences comprising external experts whose identity should remain undisclosed.

The contributions of this paper are two-folded: (1) We introduce an application independent model for managing closed groups. (2) We complement the

functional building blocks of this model with privacy and anonymity services. The building blocks of the resulting secure and anonymous multicast framework can be combined according to the application's needs. Furthermore, the interchangeability of the key management techniques allows for small to large, for almost static to very dynamic groups.

We organise our paper in four parts: We start with an overview over previous approaches for anonymity used in unicast scenarios and describe common attacks on these systems. Section 3 introduces the building blocks of a secure multicast framework and then details our design of a secure anonymous group infrastructure. We evaluate the advantages and shortcomings of our approach in Section 4 and conclude with an outlook on further work in Section 5.

2 Related Work

Related work consists of three parts. First, we present essential methods that we identified for anonymous unicast connections. Then, we look at the vulnerabilities in practical realizations. Finally, we cover work done in the area of anonymous multicast[1].

2.1 Essential Methods Used in Anonymity Schemes

We identified several essential techniques used for the anonymisation of unicast connections.

Shuffled routing minimises the risks of successful attacks from adversaries which control parts of the network. The sender chooses a set of anonymising nodes out of a cloud for relaying its message to the receiver. Under the assumption that an adversary controls only parts of the cloud, the sender has a fair chance to choose randomly one of the non-compromised nodes.

Explicit and **implicit** addresses are a means to obtain receiver anonymity, i.e. to conceal the real recipient of a message [5]: An **address is implicit** if the address contains no information either on the actual location of the addressee or on the physical reachability of the addressee. Only members of a certain group are able to use this implicit address for addressing the addressee. On the other hand, an **explicit address** contains information that can be used in a straightforward manner to route a message to the addressee.

One of the simplest techniques for anonymous communication uses such a **trusted third party (TTP) as intermediary communication partner**: A sender relays her message for the receiver to the TTP. This one replaces the sender's address with its own, and forwards it together with the original message to the receiver. Forwarding through a TTP provides sender anonymity with respect to a passive external adversary which is not situated near the sender. The former technique has the disadvantage that the TTP is a popular point of attack. The sender needs to trust it. A combination of the above technique with public key cryptography results in a mechanism called **layered encryption forwarding**: The sender adds for every intermediate hop one layer of addressing information and encrypts it for the respective hop. Each hop replaces the sender information with his own addressing information, strips off the relaying information and sends it to the next hop in the chain. Note that the layered encryption forwarding does not change the total amount of trust required in the communication relationship: All relaying hosts together are trusted with the same information on the forwarding than in the previous scheme. However, the important difference is that each on his own is not able to reveal the complete source and destination information. By adding intermediate nodes in the forwarding path, the latency is increased. However, an attacker requires substantial more control in the network, such making an attack much more expensive.

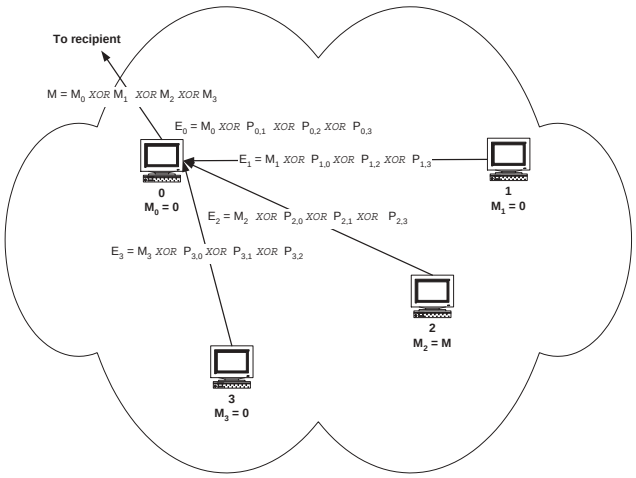


Figure 1. Superposed Sending: Example.

Superposed sending, or the **Dining Cryptographer (DC) method** provides sender anonymity by using a combination of several cryptographic transformations [6]. Three cryptographers dine at their favourite restaurant. Their waiter informs them that the bill has all ready been paid anonymously. However, the cryptographers want to know if one of them paid or if a third party paid for the diner. The solution is quite simple: Each cryptographer flips a fair coin

[1] We do not cover approaches for secure multicast as there exist several recent, good overview papers [3, 4].

with the person to his left, and another to his right. Then each cryptographer announces the XOR of the two coins he sees. If he paid the bill, he announces the opposite of the XOR. If no one paid, then the XOR of all the announced values will be 0. If someone paid, then the XOR will be 1. In a network, this solution translates as shown in Figure 1. The major drawback of the DC method is that an agreement protocol is needed for determining who is allowed to send next. This problem is called the **collusion resolution** problem. Furthermore, any three nodes can conspire and determine the sender of the message (the fourth remaining node in our case). In general, a DC network consisting of n nodes is resistant to attacks of any subset of $n-2$ nodes: Supposing that a subset of $n-2$ nodes can successfully attack such a network is equivalent in saying that the adversary is able to determine the inputs to the one-time pad shared between the two parties that do not participate in the attack. However a single one-time pad is enough to make a transmission illegible to any outsider. The superposed sending method is a sender anonymity providing technique which has been proven to yield information theoretic secrecy in the sense of Shannon. However, the usability is restricted by the collusion resolution problem and the low performance in a widely distributed anonymising network.

Mixing [7] describes a technique that combines several methods: (a) A mix consists of a cascade of untrusted third parties using layered encryption with random bit strings. (b) It supports source routing in both directions, i.e. a sender may include a return path in the same way it includes a forwarding path. (c) It fragments or pads the message sent in order to sent out only packets of same length. So, the adversary can not use the packet length as one information source in a traffic analysis attack. (d) It assembles and reorders messages in batches in order to conceal more useful information from an observing adversary. The mix technique was introduced for the email system. Several existing system relay on it. A major improvement of the mixing technique is the introduction of dummy messages in the mix system.

2.2 Vulnerabilities in Practical Realizations

An anonymous communication system is best described by its handling of possible attacks. Therefore, we begin by defining reasonable attacks on such a system:

ATTACK 1: By analysing the lengths of messages transmitted, an adversary can correlate different connections to client-server pairs. This method is called **traffic analysis of message volume**.

ATTACK 2: If an adversary can establish a correlation between the beginning and end of a connection based on his passive observation of a link, we speak of a **timing attack**.

ATTACK 3: A third form of traffic analysis attack is the **profiling attack**. I.e. an attacker can trace users in long observation periods.

ATTACK 4: Messages in which the coding is not altered during transmission are subject to the **message coding attack**. Such messages are linkeable through pattern matching, hence traceable.

ATTACK 5: A system is vulnerable to a **collusion attack**, if a coalition of corrupt insiders is able to gather more information together than each one on its own is entitled to. The exploitation of this information can break the system's security.

ATTACK 6: In a **flooding attack** order to separate the message he is interested in. The systems subject to this attack typically rely on a kind of group anonymity, meaning that each message can only be anonymous in a group of sent messages.

We analyse existing approaches according to their abilities to cope with the attacks. The practical realizations are classified according to the targeted service into **e-mail** based ones, **web** centric ones, and **network** based ones.

Approaches / Attacks	Remailer		
	Type 0	**Type 1**	**Type 2**
Message Volume	×	×	√
Timing	×	×	(√)[1]
Profiling	×	×	(√)[2]
Message Coding	×	√	√
Collusion (Insider Attack)	×	(√)[3]	√[4]
Flooding	×	×	×

Table 1. How do existing e-mail based approaches cope with different attacks?

[1] Not all mixmaster include dummy traffic to prevent timing attacks.

[2] Depends on amount of dummy traffic.

[3] If the chosen remailer is compromised, the system's security is broken.

[4] $n-1$ of n mixmasters may collude.

Attacks \ Approaches	Anonymizer[1]	LPWA [8]	Crowds [9]	Rewebbers [10]	JANUS [11]
Message Volume	×	×	×	×	×
Timing	×	×	$(\surd)^2$	×	×
Profiling	×	×	×	×	×
Message Coding	×[3]	$\surd$	**Insider:** × **Outsider:** $\surd$	$\surd$	$\surd$
Collusion	n.a.[4]	n.a.	×	×	×
Flooding	×	×	×	n.a.	n.a.

Table 2. How do existing WWW based approaches cope with different attacks?

[1] `http://www.anonymizer.com`

[2] HTTP requests frequently appear in bursts. I.e., typically, the web browser is the first to send a request. 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.

[3] With link encryption between client and proxy: weak protection, otherwise no protection.

[4] Centralised system.

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[2]. (3) Mixmasters[3] or **Type 2 remailers** are more resistant against spamming and traffic analysis attacks. Their vulnerabilities are shown in Table 1. A $\surd$ means that the discussed approach is resistant against this kind of attack, a × indicates that the approach has no protection mechanism against this attack.

Since the WWW is probably the most frequently used application on the Internet, the demand for anonymous Web browsing is increasing. We choose five representatives of this category and summarised their vulnerabilities in Table 2. They are especially vulnerable against traffic analysis of message volume and timing because they do not include any kind of delay mechanisms. A more detailed description of the approaches is given in [12].

Most approaches are concerned about a certain application, typically e-mail or WWW. **Onion routing** and the **Freedom Network** are some of the few solutions providing anonymous connections, independently of the actual application.

The Onion routing network [13] consists of a number of *onion routers*, i.e. routers that use the forwarding

[2] `http://anon.efga.org/Remailers/TypeIList`

[3] `ftp://utopia.hacktic.nl/pub/crypto/remailer/`

Attacks \ Approaches	Onion Routing	Freedom Network
Message Volume	**Endpts:** × **Between Onion Routers:** $\surd$	**Endpts:** × **Between AIPs:** $\surd$
Timing	**Endpts:** × **Between Onion Routers:** $\surd$	**Endpts:** × **Between AIPs:** $\surd$
Profiling	×	×
Message Coding	$\surd$	$\surd$
Collusion	$\surd$[1]	×
Flooding	×	×

Table 3. How do existing network based approaches cope with different attacks?

[1] resists up to $n-1$ colluding onion routers (of n existing ones in total) since n nodes are needed to decode the onion.

techniques with a cascade of untrusted third parties. These routers maintain a set of encrypted TCP connections to each others. Therefore, each pair shares one symmetric secret key. However, the core of the system is the onion itself, i.e. the 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 routing of the data). After the setup of the path, the onion proxy accepts data from the application, breaks

it in blocks of fixed length, encrypts it according to the chosen path, and transmits the result to the onion router network. On the path, each onion router strips its envelope and forwards it to the next hop. The responder proxy assembles the data blocks and sends them to the receiving application. Besides being transmitted in uniformly sized blocks, the data is mixed, i.e. collected and reordered randomly. Synthetically generated traffic may be added to long term connections. 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.

The Freedom Network [14] is an anonymising overlay network running on IP. Its similarities to Onion Routing and PipeNet are strong: The Anonymous Internet Proxies (AIP) form the backbone of the anonymous network. The anonymising protocol uses layered encryption between AIPs considered as semi-trusted third parties. The integration of anonymous mail servers using pseudonym servers is planed. However, the Freedom Network suffers from a far worse design flaw: Active attackers incorporating two AIPs can trace everyone who uses them as first and last AIP. The first AIP must mangle the payload of the incoming packet and the last one recognises them by checking if the computed checksum matches the indicated one. If it does not, the packets were mangled and an IP address can be associated to the pseudonym used so by revealing the anonymised identities.

2.3 Anonymous Multicast

Concerning anonymity in multicast, few work has been done so far. [15] introduces a simple protocol for multicast based anonymity, i.e. the protocol uses the inherent anonymity in multicast to achieve receiver anonymity and uses a set of trusted forwarding nodes to guarantee – under certain conditions – sender anonymity. However, the protocol uses the same technique as one uses for denial-of-service attacks.

3 Secure Anonymous Multicast Communication

A group typically consists of participants that send and receive data. Each participant may play both roles, e.g. in an Internet based teaching scheme, the teacher may give lectures, but may also listen to the questions asked by students. Secure multicast scenarios require the data to be sent to and received only by legitimate participants or members. In our Internet learning scenario, only enrolled students should attend the lecture. As the sender cannot control the membership in an IP multicast group, anybody may receive the multicasted data. The typical approach to solve this problem is to encrypt the multicasted data with a symmetric session key, and to distribute this key to the legitimate receivers. The distribution of the session key must be efficient in dynamic groups[4] and scalable to a large number of members. In the remaining of this section, we will now first describe the building blocks of the secure multicast framework we designed (Section 3.1). Then, we explain the extensions of this framework for anonymous but yet secure group communication (Section 3.2). Finally, in Section 3.3, we show how to deal with degenerated cases of all members staying anonymous and how the delay of distribution to anonymous users can be minimised.

3.1 A Framework for Secure Multicast

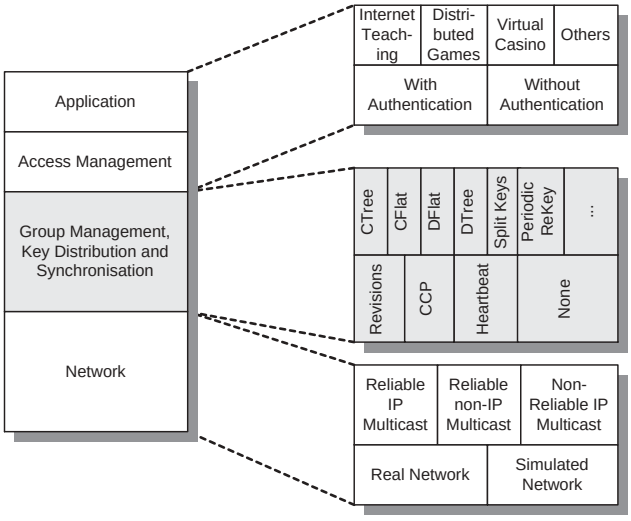


Figure 2. Secure Multicast Framework.

Our framework consists of four functional building blocks as depicted in Figure 2. We describe the tasks of each block and enumerate example representatives.

The **application** block encompasses all sort of user application programs involving more than one participant wishing to communicate. Example applications are the already depicted Internet learning scenario, distributed multi-party games, virtual casinos, the MBone applications, or Internet based trading communities.

[4] I.e. groups with changing memberships.

The **access management** controls if the joining participant is entitled to do so, i.e. if the credentials he presented upon join are valid for the requested group. Further on, it must be ensured that, at each point in time, the access rights or security policies are fulfilled (as we expect these to be dynamic, to change over time). Different admission schemes are possible and implemented: everyone, only paying participants or access lists' schemes.

The third building block called **creation, distribution and synchronisation of the necessary key material** ensures the core business of the **secure** multicast communication. Its responsibility is to provide each participant just in time with the correct keys needed to understand and / or verify the origin of multicasted data, and to send confidential and / or authenticated data to the group on the other hand. Solutions for these tasks include the following ones: (1) Early developed, centralised approaches use a key distribution centre controlling the entire distribution process [16]. (2) Recent distributed approaches rely on tree based hierarchies to manage the keys. Examples are the one way function trees [17] and rekeying approaches [18]. Sometimes, they are combined with the Diffie-Hellman types of key exchanges for performance reasons [19]. (3) Secure group management schemes such as the VersaKey family [20] include centralised to distributed approaches that may be switched during operation. Synchronisation protocols are needed in order to enable the timely delivery of the keys, and allow each participant to keep track of valid and invalid keys, too. An example of a synchronisation protocol is the revision concept used in VersaKey.

Finally, the **network** used may be a real one, a simulated one or an overlay net, e.g. virtual network. Its task is the provision of accurate multicast support for the chosen use case.

With this framework, we have a powerful basis for the development of new secure group communication protocols allowing us to exchange protocols in one block without the need to replace and customise all other blocks. One example of a possible usage is its conceptual extension to anonymous secure multicast as shown below in Section 3.2.

3.2 Design

Conceptually, we let new members join the secure multicast group without having them to reveal their identity to any outsiders and / or members of the group. Therefore, we introduce a new participant in the multicast group: the secure and anonymous multicast (SAM) server. Figure 3 depicts an example of two SAM servers A and B joining the secure multicast

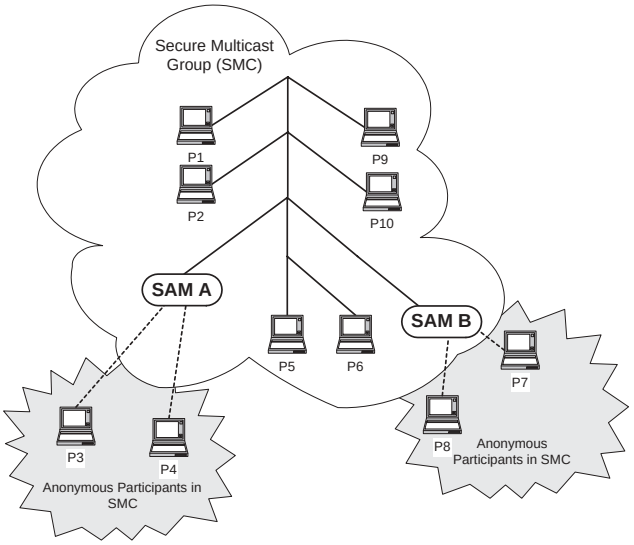


Figure 3. Illustration of our Approach.

group M consisting of the participants P_1, P_2, P_5, P_6, P_9, and P_{10}. Through the membership of the SAM server A, P_3 and P_4 participate anonymously in the group. Similarly P_7 and P_8 join the group M together with the SAM server B. All multicast traffic divulged in the group M is transmitted in a secure way by the SAM servers to their respective anonymous subscribers. So, in this scenario, the membership of the anonymous participants is hidden to external eavesdropper and internal members of the group M.

Our concept for secure and anonymous multicast communication extends the more general framework for secure group communication as shown in Figure 4. Therefore, we performed the following additions:

- The extension of the application block let to a division into two sublayers. As the application should not become dependent of the underlying infrastructure[5], the framework dependent part was decoupled from it. The resulting application programming interface (SAM-API) implements the application specific functions related to the interfacing of application and framework. For example an agents' application requires authenticated and confidential communication between the agents "in the field" bidding for network resources. Hence, the agents' application needs an interface to the framework to require the fulfilment of this demand. A distributed game, on the other hand, may only require, if at all, the authentication of a player in the join phase and no con-

[5] It seemed a poor design choice to do so just for anonymity purposes.

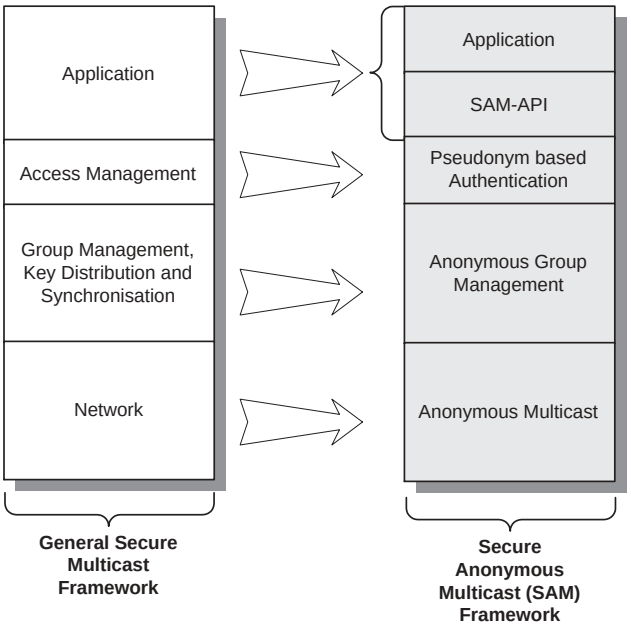


Figure 4. A Framework for Secure and Anonymous Multicast (SAM).

fidential data transmission at all. In the Internet learning scenario, finally, only the data transmitted during examinations must be authentic.

- The access management block is enhanced with a pseudonym based authentication mechanism: Each user is given a randomly generated pseudonym upon verification of the credentials he presented during the authentication process.

- The management of the group keys used for data traffic encryption and for confidential and authenticated distribution of the group management information is done by standard secure group techniques or methods that draw profit of the group topology such as the one-to-many scheme described in [21]. The Internet learning application, e.g., uses a one-to-many scheme that scales to large, dynamic groups for lectures and a many-to-few scheme for written exams. Furthermore, we need to include pseudonymity awareness in the addressing schemes used for members.

- Finally, we need to add the required network infrastructure – the SAM servers – allowing for users to remain unknown to outsiders or to both outsiders and the group.

The SAM servers joining on behalf of the users that 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 anonymised group members, receive the secure group communication through the SAM server. Therefore, we use the SNAP (*Secure Non local Anonymising Protocol*) architecture described in [12]. 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 protected transmission between SAM server and user.

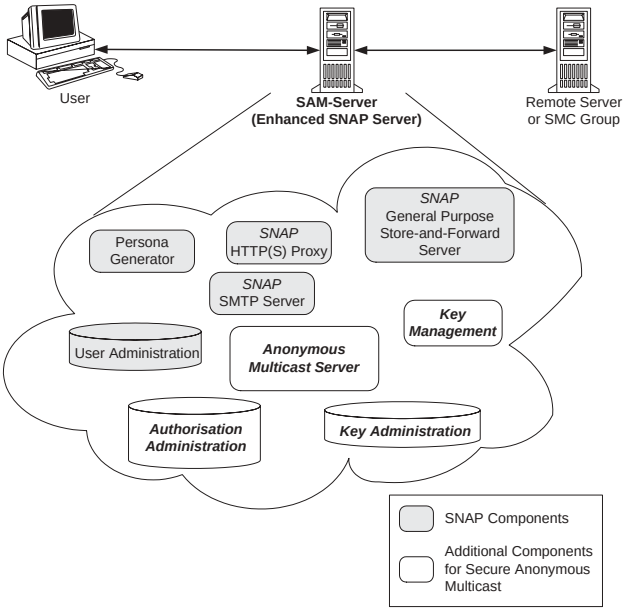


Figure 5. Enhanced SNAP Server for Secure and Anonymous Multicast.

The implications of the usage of the SNAP architecture on the simplified secure multicast framework are many-folded: First, the new SAM server derived from the SNAP server must be able to join a secure multicast group. Therefore, we included a key management component in our design and implementation (see Figure 5). Its task is to organise the keys used in every secure multicast groups it participates in. Second, we need to administrate the keys and credentials given by the SAM user to the server to join a group. This includes the administration of the keys and the authorisation credentials. Finally, the SAM server should distribute the received multicasted data to the entitled users and send the user's data encrypted and/or

signed with the correct key(s) to the targeted secure multicast group.

3.3 Special Cases

3.3.1 Degenerated Cases

We distinguish between two degenerated cases such as depicted in Figure 6: The first only includes anonymous participants in the secure multicast group without any normal members. The second one includes only normal users. This case is equivalent to the case of a secure multicast group with the exception that no member of the group can claim to know that there are only normal members. Only a collusion of all members could reveal this fact. As long as there is one member not colluding, he could be assumed to be a SAM server.

Both cases do not require a change in the building blocks. They are handled as "normal" distributed scenarios.

3.3.2 Merge of Unicast Connections

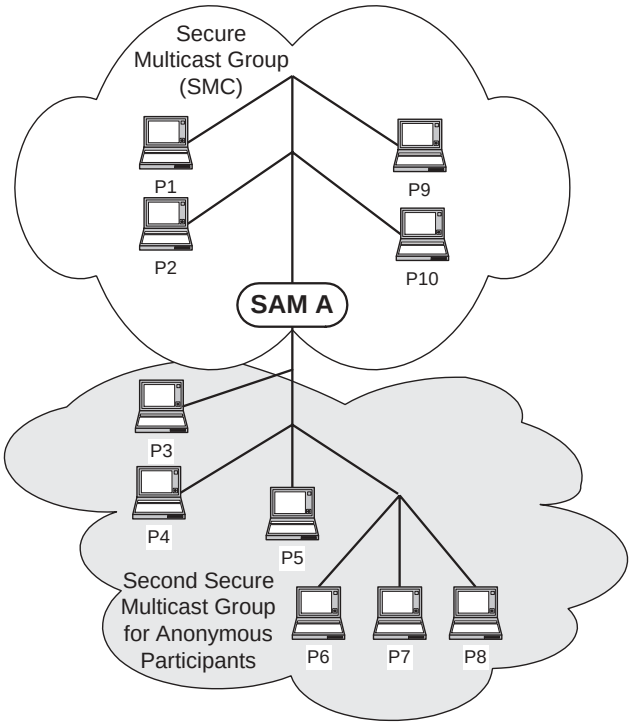


Figure 7. Merge of Unicast Connections.

If a SAM server reaches a critical number of users subscribed through it to the same secure multicast group, it may include all these users in a second secure multicast group as depicted in Figure 7. Of course, this second group will only include users that are have at their disposal the required technology to join a secure multicast group. For this case, we extended the anonymous multicast server in the SAM server with a dummy traffic generating engine as commonly used in mixes (see Section 2 for a description of the mix technique).

3.3.3 Certifications

A special access management scheme was included in the order to allow for fair, semi-blinded examinations. We introduce certified pseudonyms for students. The protocol for a student *S* who desires to take a semi-blinded examination of examiner *E* is as follows:

1. *S* checks if *E* allows for semi-blinded examinations in the digitally signed course description. In case of a negative answer, semi-blinded examinations are not possible for this course.
2. In the case of a positive answer, *S* requests a certification of his pseudonym *P* used in the course from the certification authority of the university. Therefore, he needs to proof that the pseudonym *P* is bound to his real identity in an unalterable manner by his SAM server. The certification authority stores the certificate in a safe fashion and puts the pseudonym together with the timing information of the exam on the access list for the secure multicast group of the examination.
3. *S* joins this group anonymously through the indicated SAM server at the time of the exam.
4. *E* grades the signed performance of *P* and forwards the grade to the certification authority and to the pseudonymous user *P*.

4 Evaluation

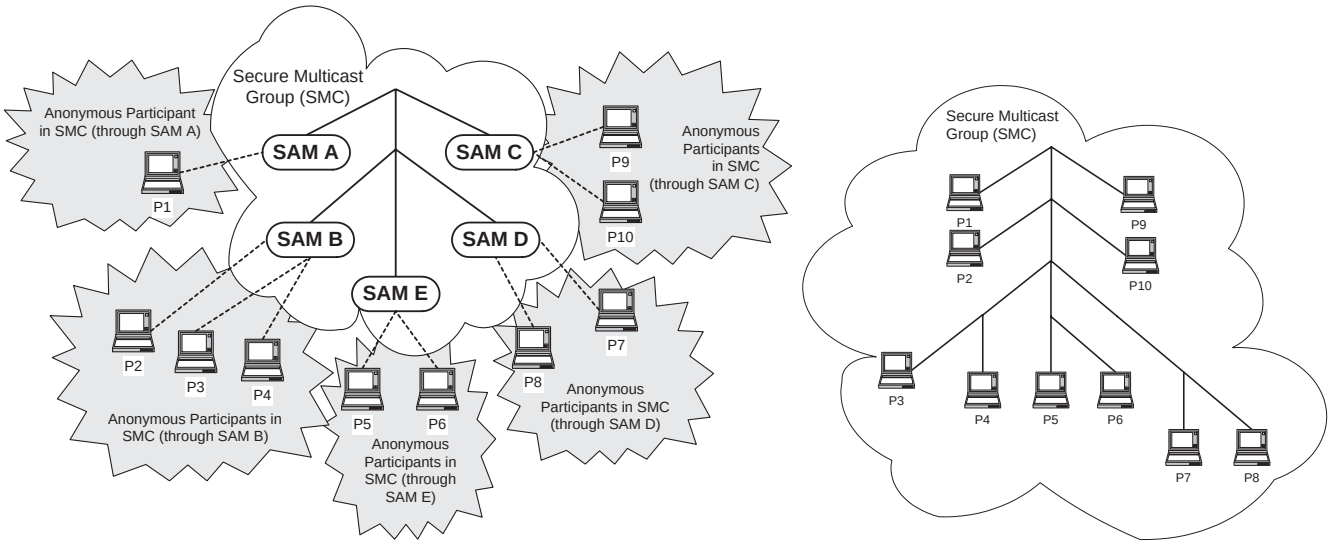
4.1 Scalability of the SAM framework

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. Therefore, techniques such as Onion Routing or the Freedom network (Section 2) are employed.

4.2 Complexity Analysis

The additional costs for the SAM architecture originate from two improvements on a typical group architecture: (1) the introduction of anonymity mechanisms, and (2) the usage of secure group communication.

The latter one is heavily dependent on the used group key management scheme. The communication costs of the most frequent operations in centralised



(a) Only Anonymous Participants.

(b) No Anonymous Participants.

Figure 6. Degenerated Cases.

and distributed, tree-based approaches are shown in Table 4. n denotes the number of members in the group, k the length of the key in bits.

Operation	**Centralised**	**Distributed, Tree-based**
Group initialisation	nk	$2nk + log(n)$
Join of a member	$nk + log(n)$	$2log(n)k + log(n)$
Leave of a member	$nk + log(n)$	$2log(n)k + log(n)$

Table 4. Communication costs (in terms of multicast size) of the most frequent operations in common secure multicast approaches.

The impact of the anonymisation depends on both the number of users per SAM server and the total number of SAM servers. The first factor gives an indication for the number of additional unicast sessions needed. The second factor specifies the number of additional joins in the secure multicast group. So, the mechanisms in each building block of the SAM framework for a specific application should be carefully chosen with respect to the expected group topology.

4.3 Resistance to Attacks

The resistance of the SAM framework to the defined attacks cannot be given at a general level because it depends on the exact configuration used.

A near ideal setup uses SAM servers organised as in the onion routing approach with pseudonym based authentication and dummy traffic generation. The SAM servers will not only process multicast traffic, but also the traditional traffic of the SNAP server: e-mails, web browsing and other TCP based traffic.

5 Conclusions

In summary, the SAM framework provides an environment for anonymous group communication derived from a general purpose and application independent secure multicast framework 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.

Furthermore, we prevent observers in the local environment of the user from learning any information about the traffic transmitted between SAM server and

user. This property called *local anonymity* is inherited from the SNAP server.

Finally, the SAM framework allows for *hybrid authentication* mechanisms. Some application require only registered users to participate, but after authentication, they have no interest in which particular user send which message. An example of such an application is a virtual casino: A player must reveal some personal identification such as age and financial information for legal and operational reasons when he changes money to playing chips. But the information on which game he wins his chips should not be accessible. In other words, once he got the chips, he should remain anonymous in the virtual casino. The simplest solution for this approach in the SAM framework is a pseudonym based solution.

Further research on this topic will encompass (1) the analysis of the behaviour of different applications in the SAM framework with respect to selected authentication methods and group management techniques, (2) the quantitative evaluation of the different components concerning performance and usability, and (3) the theoretical assessment of the degrees of anonymity resp. pseudonymity achieved with respect to the defined attacks.

References

[1] W. Fenner, "Internet group management protocol, version 2," RFC 2236, November 1997.

[2] Anton Ballardie and John Crowcroft, "Multicast-specific security threats and counter-measures," in *Proceedings of ISOC Symposium on Network and Distributed System Security*, San Diego, CA, USA, February 1995.

[3] Lakshminath R. Dondeti, Sarit Mukherjee, and Ashok Samal, "Survey and comparison of secure group communication protocols," 2000.

[4] Matthew J. Moyer, Josyula R. Rao, and Pankaj Rohatgi, "A survey of security issues in multicast communications," *IEEE Network*, November / December 1999.

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

[6] David Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, pp. 65–75, 1988.

[7] David L. Chaum, "Untraceable electronic mail, return adresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, February 1981.

[8] 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.

[9] Michael K. Reiter and Aviel D. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, November 1998.

[10] Ian Goldberg and David Wagner, "TAZ Servers and the Rewebber Network: Enabling Anonymous Publishing on the World Wide Web," *First Monday*, vol. 3, no. 4, April 1998.

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

[12] Nathalie Weiler and Bernhard Plattner, "Secure Anonymous Protocols for Local and Multicast Environments," Technical Report 73, TIK, ETH Zürich, Switzerland, October 2000.

[13] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag, "Anonymous connections and onion routing," *Journal on Selected Areas in Communications*, vol. 16, no. 4, May 1998.

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

[15] Clay Shields and Brian N. Levine, "A protocol for anonymous communicastion over the internet," in *Proceedings of the 7th ACM Conference on Computer and Communication Security (CCS'2000)*, Athens, Greece, November 2000.

[16] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proceedings of ACM SIGCOMM '97*, Cannes, France, September 1997, pp. 277–288.

[17] David Balenson and David McGrew amd Alan T. Sherman, "Key management for large dynamic groups: One-way function trees and amortized initialization," Internet Draft draft-irtf-smug-groupkeymagmt-oft-00.txt, August 2000.

[18] Ohad Rodeh, Kenneth P. Birman, and Danny Dolev, "Optimized group rekey for group communication systems," in *Proceedings of Network and Distributed System Security Symposium (NDSS'00)*, San Diego, CA, USA, February 2000.

[19] Yongdae Kim, Adrian Perrig, and Gene Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *7th ACM Conference on Computer and Communication Security*, November 2000.

[20] Germano Caronni, Dan Sun, Marcel Waldvogel, Nathalie Weiler, and Bernhard Plattner, "The VersaKey Framework: Versatile Group Key Management," *IEEE Journal on Selected Areas in Communications, Special Issue on Middleware*, September 1999.

[21] Nathalie Weiler, "SEMSOMM - A Scalable Multiple Encryption Scheme for One-To-Many Multicast," in *Proceedings of the IEEE 10th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '01)*, June 2001.