Preventing Traffic Analysis in Secure Group Communication*

Pavan Verma and Atul Prakash
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor
{*pverma,aprakash*}*@eecs.umich.edu*

Abstract

Group security protocols have primarily focused on the three major security problems – authentication, confidentiality and integrity – and overall these problems have been well addressed. However, the problem of traffic analysis where an attacker monitors messages and is able to obtain useful information from the protocol header fields as well as the size, number or time of the packets has not been given much attention. In this paper, we discuss these problems from a point of view of anonymity – privacy of the group's identity and membership. We present modifications to secure group communication protocols that provide anonymity to groups. Our scheme lets only members of group g know whether or not a packet is for group g. We also present an authenticated and private key-recovery scheme that allows only group members to detect missed rekey messages from subsequent group messages. We discuss the security guarantees our solutions provide, analyze the performance costs and discuss several performance optimizations. We show that adequate performance can be achieved for many applications, including protecting traffic on broadcast-based networks (such as Ethernets) from traffic analysis.

1 Introduction

The last few years have been a period of substantial research in the area of group communication systems. A lot of this research has addressed security issues and this has led to several solutions for the three major security problems – user and data authentication, data confidentiality and data integrity – in the setting of group communication. However, the problem of traffic analysis on group protocols has not been studied in much detail. Traffic analysis is the monitoring and analysis of messages to gain useful information about the communicating parties and the messages they are exchanging. Even if the payload in the packets is encrypted, protocol header fields, size of the packet, time at which it was sent, etc. can reveal useful information.

Secure group communication protocols are usually designed to protect the payloads rather than prevent analysis based on header fields (for example, group identifier), quantity of messages sent to the group, sender's identity, membership changes, etc. However, by analyzing such data, an adversary can ascertain things like membership of the group, roles of the members within the group, communications activity level of the group, etc.

*This work is supported in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0508, the National Science Foundation under grant 0082851, and gifts from IBM, Intel, and Microsoft. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and do not necessarily represent the official policies or endorsements, either expressed or implied, of research sponsors, including the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

One of the things traffic analysis can be used for is to compromise the privacy of a group by gaining information about group membership, which packets belong to the group, etc. This leads us to the issue of anonymous communication for a group. Ideally, anybody who is not member of the group should not be able to get any knowledge about the group. More realistically, the information that a non-member can get should be minimized.

The issue of anonymity is very important in certain situations. Sometimes, preserving anonymity of a group and it's members could be as important as confidentiality of the data they are exchanging. The work of intelligence agencies (such as CIA, FBI) is an example where anonymity is really important. If the identity of informers is not completely safeguarded, there might be danger to their lives.

In this paper, we focus on trying to achieve anonymity in the setting of group communication. Specifically, the type of anonymity we look for is *unlinkability* of group messages. This is the property by which two group messages cannot be correlated – they cannot be said to belong to the same group. For this, we present a technique that reveals the identity of the group to which a message belongs to only members of that group. That is, if a message belongs to group g, only members of g can know that the message is for g. In this way, anyone who is not a member of the group cannot know which group the message belongs to, thus providing anonymity to the group.

A second thing that traffic analysis can achieve is to help in attacking the protocol or making the attack easier. Group protocol messages often give away information that is useful for designing potential attacks that don't necessarily violate the main guarantees of a security protocol (authenticity, confidentiality and integrity), but do cause the protocol to incur significant cost in defending against the attacks – potentially higher than required to mount an attack. For example, delivery of key distribution messages can be attacked if an adversary can distinguish between key distribution messages and other types of messages. Alternatively, an adversary may be able to inject cleverly-designed messages that cause the members to suspect that they do not have the latest key or are no longer part of a group. If a server is used to handle member joins or to recover lost keys, an adversary could leverage the protocol to mount denial-of-service attacks on the server by causing all members to simultaneously request something from it. The main point is that if an adversary can monitor even meta-information in packets to determine a protocol's state, it can reduce the amount of work it has to do to attack a protocol. A similar behavior is observed in other attacks on computer systems – where systems are first probed by network scanning tools and then they are selectively attacked based on observed vulnerabilities.

This paper makes the following contributions. It presents a modification to group communication protocols such that someone who is not a member cannot know which group a message is for. If there are several groups in the system, this provides sufficient anonymity to the group. We analyze the security of our solution and its effect on message throughput and processing time. Our findings indicate that in typical scenarios, the throughputs are likely to be reasonable for most applications (range of 1-10 MBps[1] for current hardware). And where higher data throughputs are required, the techniques are amenable to use of hardware accelerators.

We discuss potential applicability of our techniques for preventing traffic analysis within a broadcast-based communication medium such as Ethernets. Although an Ethernet is not traditionally thought of as a group communication system, all the communication within the Ethernet can be modeled as either two-party group communication or a broadcast to all hosts on the Ethernet. Our solutions can potentially be applied at the MAC-layer in Ethernets to prevent network-level traffic-analysis based on MAC addresses or IP header fields.

We also discuss problems with key distribution protocols for group communication and present a key recovery protocol. The protocol allows group members to infer from the subsequent group messages that they missed a key distribution message. The inference step is authenticated and also private – only group members can infer that the group has changed keys.

[1] MBps=mega bytes per second, Mbps=mega bits per second

The paper is organized as follows. Section 2 briefly describes fundamentals of secure group communication, presents a sample group protocol message and discusses our communication and threat models. Section 3 describes the problems of traffic analysis and secure key recovery in group communication. Section 4 surveys relevant related work and discusses why none of the existing solutions are suitable. Section 5 presents our proposed traffic analysis resistant message construction, analyzes its security and performance overhead and discusses its application to preventing traffic analysis in broadcast mediums such as Ethernets. In Section 6 we present a secure key recovery protocol, analyze the security it provides and the performance overhead it incurs. In Section 7, we conclude the paper and discuss some opportunities for future work.

2 Background

In this section, we introduce relevant background about secure group communication systems. We also introduce notation, describe a sample message construction which we will use to describe the problems. We also discuss our communication and threat models.

2.1 Group Keys and Views

The goal of a secure group communication system is to provide infrastructure that allows a group of entities to communicate securely. Of course, to be widely usable it should be scalable, flexible and efficient. Several secure group communication systems have been proposed [17, 1, 27], with different security and fault-tolerance guarantees, key establishment mechanisms and other aspects such as provisioning issues.

However, the basic idea in all group communication is the same. All group members possess a symmetric key (such as `AES, Blowfish, DES, 3DES`) called the *group key*. The group key is known only to group members and is used to provide authenticity, confidentiality and integrity guarantees of data. We note that the security that can be provided using such a design is limited in granularity to the group rather than the individual. Group members might directly collude with an adversary by giving it the group key or could spoof other group members. Thus, the assumption that group members is necessary.

What complicates the situation is that the group membership could be dynamic – old members can leave the group and new members can be added into the group. There is the notion of a *membership view*[2] which is a period during which the group membership does not change. To maintain confidentiality of group data in such a dynamic membership scenario, the group key should be known *only to current* group members. Thus, whenever the group membership changes, the group key is changed and the new key is distributed to all group members. This process of changing the group key is called *rekeying*. Key management is therefore a central part of any secure group communication system. Several different schemes can be used for key management and broadly two classes of solutions exist – those based on server-based key distribution protocols (e.g., centralized key distribution and logical key hierarchies [34, 33]) and those based on key agreement protocols such as CLIQUES [32] that typically rely on group Diffie-Hellman techniques. Key distribution protocols are more scalable but they rely on trusted servers to be available to distribute the keys. In this paper, we assume that a key distribution protocol is used in the group, though the specific protocol used is not important.

2.2 Notation Used

We use the following notation when presenting group protocol messages:

[2] also referred to as *group view* or simply the *view*

g	group identifier
v	membership view identifier
$\{m\}_k$	m encrypted with key k
$h(m)$	secure hash (such as MD5, SHA1) over m
$H_k\{m\}$	MAC (HMAC or any other keyed MAC) of m with key k
k_v^g	group key for group g in view v
N	nonce

Some additional notation will be introduced as and when required.

2.3 Sample Protocol

Group Identifier	View Identifier	Encrypted Payload	MAC over entire message
g	v	$\{payload\}_{k_v^g}$	$H_{k_v^g}\{g, v, \{payload\}_{k_v^g}\}$

Figure 1: Sample Group Message

Figure 1 shows a sample, general group protocol message. The message contains four fields: group identifier, view identifier, encrypted payload and MAC. The group and view identifier fields are required by a receiver to associate the message with the correct group and the correct view.

Group 1	**Key 1**
Group 2	**Key 2**
Group 3	**Key 3**
...	**...**
Group n	**Key n**

Figure 2: List maintained at each user

Each user in the system maintains a list whose entries are <group identifier, group key> tuples as shown in Figure 2. When a message is received, the group identifier from the message is used as an index into this table. If the entry does not exist, the message is simply discarded. If the entry exists, the receiver gets the group key, verifies the MAC check and then proceeds processing the message as with any other protocol.

In addition to the group and view identifier fields, a message might also contain additional group protocol header fields. However, it is not necessary to have them because they could achieve their purpose by being part of the payload itself. For this reason, they are not shown in Figure 1.

2.4 Communication Model

The communication environment we consider consists of users (individuals) and groups. Each group could have several members and each member could belong to several groups. Figure 3 graphically presents the membership model. Ovals are used to represent groups and squares to represent users.

We assume that the groups use multicast to send packets to the whole group. However, the group's control servers could use unicast also to communicate with members. Different groups might use the same or different multicast addresses. They could also use the same or different ports.

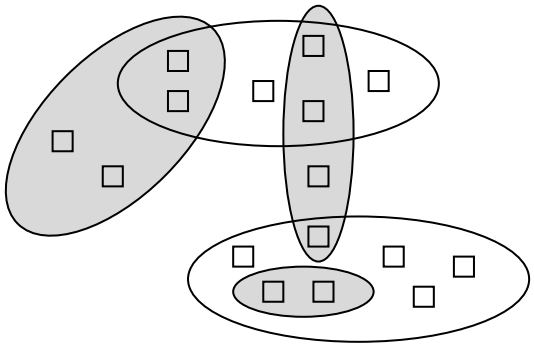

Figure 3: Groups and users in the system

2.5 Threat Model

In our model, the adversary is an eavesdropper who can monitor any packet as it is sent over the network. It can also monitor packets at more than one point in the network. An adversary that can monitor packets at multiple points is supposed to be stronger than an adversary that can monitor packets at only a single point. More generally, we assume that among two adversaries, the one that can monitor packets at more points is stronger. Apart from this, we do not further quantify the strength of an adversary. The adversary can also actively insert packets into the network.

We assume the group communication protocol to successfully provide authentication, confidentiality and in integrity. Our adversary cannot break these guarantees. Thus, it cannot get any keying material used by the group members or break any of the crypto algorithms used. We also assume that group members are honest while they are member of the group. That is, they do not do things like handing over the group key to an adversary.

2.6 Security Analysis of Sample Message

The message shown in Figure 1 provides guarantees of confidentiality, integrity and authenticity. Confidentiality is provided by encrypting the payload with the group key which is known only to current group members. Integrity is provided by including a MAC over the entire message. Authenticity is also provided by the MAC field but its guarantee is limited because it only promises that some current group member sent the message; it cannot guarantee which member sent it.

Any secure group communication system would also have authentication mechanisms while admitting a member into a group (for example, using public key infrastructure [26, 18], KERBEROS [31, 20] based, etc.). These combined with the security guarantees discussed above provide a pretty comprehensive solution to the three traditional security problems of confidentiality, integrity and authenticity for the problem of group communication.

3 Traffic Analysis

In this section we discuss the problems of traffic analysis and key recovery as they apply to secure group protocols. We start by showing how the sample message of Figure 1 is vulnerable to traffic analysis. Then, we discuss the need for a secure key recovery protocol, again with reference to the message of Figure 1.

3.1 Group Anonymity

Although the message shown in Section 2.3 provides confidentiality, authenticity and integrity of messages, it gives away the identity of the group which might be undesirable in high security environments such as intelligence work, battle situations, etc.

The group and view identifier fields are sent in the clear in each message. This lets anybody (even non-members) with access to the packet know which group the packet belongs to. By observing the group identifier in any two group messages, it can be determined whether they belong to the same group. This can be further combined with observation of packets close to different entities to reveal whether they belong to the same group. This might be considered a breach of privacy in some situations and therefore, these fields should be kept secret if possible. However, doing so it not straightforward because the only secret we have to work with is the group key, but as discussed in Section 2.3 the group and view identifiers themselves are required to know the correct group key for the message.

As mentioned in Section 2.3, the protocol header might send many more fields in the clear. Doing so can give away very important information about the working of the group. For example, a typical field in a group protocol's header is the type of message field. This tells whether the message is a data message, key distribution message, fault detection message, etc. Traffic analysis based on such header fields can be used to figure out things such as the frequency of different types of messages, temporal relationships between types of messages (for example, are some kind of messages always preceded or succeeded by other types of message), roles and importance of members within the group, etc. This can be used to mount more potent attacks or to reduce the work the attacker has to do to carry out an attack. For example, if an attacker can predict when control messages (such as key distribution messages) are sent out, it can selectively block them to cause severe disruption to the group's functioning. Traffic analysis can thus help the attacker cause severe damage to the group with only a small amount of effort. Hiding such header fields is normally straightforward because they can be a made a part of the encrypted payload itself and thus we don't consider it further.

3.2 Key Recovery

As mentioned in Section 2.1, key management is an important part of any secure group communication system. A new key needs to be generated and distributed whenever the group membership changes. This could be done using a centralized or distributed scheme. In centralized schemes, a single server would generate the new key which is then distributed to the members by the key generator directly or by using other key distribution techniques such as LKH [34, 33]. In either case, key encrypting keys are used to guarantee authenticity, confidentiality and integrity of the key distribution messages. In a distributed key generation scheme many members participate in generating the new key using group Diffie-Hellman based techniques such as in [32].

Efficiency and scalability are very important issues in group communication systems because the group could be very large, highly dynamic and require high throughput. Therefore, using a reliable transport is not always viable. Thus, when the new key is being distributed or the protocol for distributed key generation is being followed, messages could get lost. Even if a reliable transport is used, an attacker could selectively disrupt these key distribution/generation messages. The effect of message loss on a group member is that it will not have the new key and will thus not be able to receive or send messages in the next view. This is a serious situation, especially if we think of it in terms of an attack: an attacker needs to disrupt only a few key distribution/generation messages in order to disrupt communication for the period of an entire view. A group communication system could use one of two strategies to deal with such attacks during rekeying:

1. Ensure that all members receive a key before proceeding, removing members that cannot be rekeyed. The removed members would have to rejoin the group when they discover that they have been removed from the group.

2. Not require that all members receive a key before proceeding. Instead, provide a way for members to detect from the subsequent messages that they have missed a rekey. The members can then get the new group key from a trusted server.

In either case, there is a potential for mischief by an adversary. If the adversary can make significant number of members think that they have been disconnected from the group or have missed a rekey, the protocol may result in message implosion at the admission server or the key distribution server when members attempt to either rejoin the group or recover the key.

In such a scenario, it is desirable to have a mechanism that allows members who do not have the latest key to detect this situation from the data messages and then request the new group key from the key distribution authority or some other group member. However, a simplistic scheme will not work for this.

For example, let's consider a scheme where the view identifier is a sequence number. If the current view is i, the next view will be $i + 1$. Upon receiving a message, if a member sees that the view identifier has been incremented, the member takes it to mean that the group key has changed and that it missed the corresponding rekey messages. Then it requests the new group key from some trusted server. This scheme might seem to work but it has a serious flaw: the view identifier is predictable. An attacker can simply frame a message with the view identifier equal to the current view identifier plus 1, include some garbage payload and send it to all the group members. This message would make *all* group members *think* that they do have the current group key. Note that upon receiving such a message, members would not be able to authenticate it as they would be under the impression that they do not have the current group key needed to check the MAC. Thus, believing that they need to get the new group key, all group members would send out key requests to the trusted server causing a message implosion on the it. This is a potent DoS attack if the group has many members.

The problem with the above scheme is that there was no sender authentication in the inference procedure followed by a group member to determine whether the group key changed. Another problem with it is that even non-members can determine that the group view has changed. We discuss in Section 6 that authenticating the inference procedure can be achieved by a commonly used technique called hash-chaining as described in several papers [12, 28, 15, 23]. However, making it confidential so that only group members know that the group view has changed is more challenging. The reason is that any scheme to be used must assume that the receiver might not have the latest key. Thus, the only secret we can use seems to be the previous group key. However, this cannot provide a good enough solution because new group members (those which have been added during the view transition) will not have the previous key and thus messages from them cannot be used for informing other members about rekey misses. Thus, there remains no secret to provide the inference confidentiality of the view change. We present our solution to this problem in Section 6.

4 Related Work

There has been quite a bit of work done on providing anonymity in communication such as DC-net [7], Anonymizer [3], Mix-net [6] onion-routing [9], Crowds [24] and Hordes [30]. However, all of this work has concentrated on a slightly different problem than ours. They try to provide sender anonymity, that is the property that nobody (including even the receiver) should know who the sender of a packet is. On the other hand, the problem we are addressing is the *unlinkability* of packets or group members with the group. That is, it should not be possible for non-members to determine whether or not two packets or two users belong to the same group.

DC-net [7] is a system the provides unconditional anonymity. However, it requires setting up a large number of very long keys and is very vulnerable to disruption attacks. This makes this scheme impractical in reality.

Anonymizer [3] is a single proxy that achieves anonymity by routing all traffic through it. The obvious problem with it is that the single proxy is a single point of failure as well as a single point of attack for an adversary.

The idea of a Mix-net [6] is to have a network of intermediate proxies or Mixes called the Mix-net. The traffic is routed through the Mix-net and packets are encrypted so that each Mix knows only the identity of the preceding Mix and the next Mix to which it has to send the packet. This makes the Mix-net resilient to compromise of some of the Mixes but not all. Onion-routing [9] is a technique that builds upon the ideas of a Mix-net and offers an implementation that anybody can use. However, like the Mix-net, it also uses asymmetric encryption techniques which makes it inefficient.

Crowds [24] is a system that is also based on the idea of using a network of proxies to route packets. The Crowds routers use a very simple technique to route packets: with a fixed probability p_f they forward the packet to its destination and with probability $1 - p_f$ they forward the packet to a Crowds router chosen at random. Using this technique Crowds is very efficient and resistant to compromise of some Crowds routers. Hordes [30] builds up on the idea of Crowds and enhances it by exploiting the inherent anonymity in multicast to get better efficiency. Each sender using Hordes joins a Multicast group that is used for the return traffic.

It is not apparent how to apply any of these techniques to solve the problem we are addressing. Moreover, group communication adds its own complexity as there are multiple receivers of each packet. Furthermore, all these schemes require that the packet covers at least some part of the route unencrypted. This part is normally before the packet enters the network of intermediate routers and after it exists it. Traffic analysis on the group protocol header fields can still be done in this region.

Other work on preventing traffic analysis is the use of traffic padding [11, 36, 14] or mixing/delaying traffic at the routers [6]. These schemes do not modify the payload cryptographically. Instead, messages are padded to make them equal size and messages may be delayed or extra messages sent to ensure a uniform rate, so that no useful information regarding timing of communication or size of messages between the communicating parties can be discerned. The downside of such schemes is that they cause high network traffic as well as delay messages. Also, the schemes do not prevent analysis of the group protocol fields – an adversary can still analyze them to potentially learn useful information such as membership changes on group rekey events.

IPSec [13], combined with aggregation of multiple IPSec streams between two gateways, can also limit the information available to an adversary by traffic analysis. However, this approach does not work when attackers have access to packets behind the gateway boundaries (e.g., the adversary is able to compromise one principal in the system and able to join an IPSec-based VPN). Furthermore, IPSec is a two-party protocol, and it is not apparent how to extend the ideas to multi-party communication.

Broadcast authentication protocols are also relevant to our work because of the similarity of broadcast to multicast and group communication. Tesla [23] and Biba [21] are two protocols for authenticating broadcast communication from a sender to a large group. Tesla, for example, uses authenticated hash chains to efficiently authenticate messages of a broadcast. However, these protocols by themselves are not designed to provide confidentiality and it is not apparent how to apply them in a group setting where confidentiality is also required and not only authenticity and integrity as in the broadcast setting.

5 Traffic Analysis resistant protocol

In this section, we present a construction for group messages that is resistant to traffic analysis. Our construction is based on the sample protocol message described in Section 2.3. We first give an outline of our goals. Then we discuss the issue of network/transport layer traffic analysis. We then present the new group message, analyze it's security and performance. We end the section with a discussion on applying our scheme for making Ethernet traffic resistant to traffic analysis.

5.1 Goals

In general, our goal in preventing traffic analysis is to allow someone who is not a current group member (that is, a potential adversary) know as little as possible about the group and the group members. More specifically, our goals with respect to the sample protocol message of Figure 1 are as follows:

1. only a current member of group g should be able to know that a message is for group g. By current group member, we mean someone who is member of group g in view v. To simplify presentation and for better readability, we ignore the view identifier field in this section but come back to it in Section 6.
2. group protocol header fields (apart from group and view identifiers) should not be sent in the clear as they might give away a lot of information about the group. This can be achieved in a straightforward way by making these header fields a part of the encrypted payload and thus we do not discuss it any further.

We assume that the group messages are sent over a broadcast network (e.g., satellite, Ethernet or posted at a shared site). Many groups are expected to share the same broadcast network. We do not actually require many groups to be broadcasting to the network simultaneously. However, from an adversary's perspective, a possibility should exist for any group to be sending messages to the broadcast network at any given time. We also assume that the number of groups, n, sharing a broadcast network is sufficient to prevent collusion attacks such as members from $n-1$ groups colluding to determine if the remaining group is sending a message. We do allow and expect multiple broadcast networks to exist in practice – all we require is that n be sufficiently large to prevent collusion attacks. Each user in the system is expected to be member of multiple groups so that there is a possibility that a packet sent could belong to multiple groups. We will guarantee that a non-member for a group cannot correctly associate a message with its group or determine whether two or more messages belong to the same group.

5.2 Network layer traffic analysis

Note that our primary focus is on limiting the information that can be obtained by examining information in the group protocol messages themselves, such as group identifiers, rekey events, etc. Analysis can still be done at protocol layers below the security protocol layer. However, information available to an adversary will be less precise (especially if a broadcast medium shared by multiple groups is used) and it may require more extensive monitoring across the network, increasing the effort required by an adversary.

In general, we expect complete protection against network layer traffic analysis to be very expensive. Making higher level security protocols resistant to such analysis and attacks thus becomes even more important. However, if desired, the techniques proposed in this paper can be combined with other techniques such as traffic padding to make network-level traffic analysis more difficult.

In particular, network and transport layer headers (`IP, TCP, UDP, IP multicast`, etc.) give out significant information, such as IP addresses of the group members. However, the point to emphasize is that using some scheme that makes the network/transport layers resilient to traffic analysis is not sufficient by itself. The reason is that the group protocol fields will still be sent in the clear and can thus still be used for traffic analysis. Thus, a more rigorous solution at the group protocol level is desirable.

As mentioned above, we assume a broadcast medium in which several groups are members of the same broadcast channel. Normally, this is not the case because different groups use different multicast addresses and ports for communication, allowing the adversary to do network/transport layer traffic analysis. One way to form a common broadcast medium is to have all groups use the same multicast address and port. These fields would then become useless for traffic analysis. Obviously, using such a broadcast medium would result in low throughput for individual groups because every user would receive every group's packets (even if they are not a member of that group) and might have to spend significant amount of computing power discarding irrelevant messages.

In this paper, we restrict ourselves to preventing traffic analysis on the group protocol while assuming a broadcast medium shared by several groups. The issue of how such a broadcast medium can be efficiently and practically implemented requires further research. We also pose the problem of preventing network/transport layer traffic analysis in the setting of group communication as a separate and independent problem. Section 4 does discuss some techniques to prevent network/transport layer traffic analysis.

5.3 Solution

As mentioned in Section 5.1, our main goal is to hide the group identifier so that only current group members can know which group the message belongs to. For doing so, the only secret that we have to work with is the group key. However, as described in Section 2.3, the group identifier in the message is required by the receiver to know what group key to use for that message. Thus, there is a conflict between using the group key to hide the group identifier without revealing the group identifier itself to someone who does not have the group key. The solution for doing this is not obvious and something simple like just encrypting the group identifier with the group key will not work because the receiver would just not know which group key to use to decrypt that field.

The idea of our solution is to send a MAC of the group identifier with the group key, i.e. $H_{k_v^g}\{g\}$ in place of just the group identifier g. Upon receiving such a message, the receiver associates the message with its group as follows. Remember that each user maintains a list of <group identifier, group key> pairs for each group it is member of (Figure 2). Upon receiving a message, the user iterates through this list, calculating $H_{key}\{gid\}$ for each $\langle gid, key\rangle$ pair in the list. It checks whether this value matches with the value in the message. If the value matches for group G say, then it means that the message is for group G. If there is no match for any group in the list, it means that the message is not for any group the receiver is member of and thus the message is discarded.

A problem with the above scheme is that within a view, the group key is the same and therefore, all messages (of the same group) within a view would have the identical MAC field. This would allow anybody with access to the packets to check if two messages belong to the same view and thus the same group. Therefore, we need to ensure "freshness" of the MAC field for each messagefootnotehere *freshness* means that the field is not repeated for two messages.. For this, a per message nonce is used for salting purposes. Thus, the final solution is to send $N, H_{k_v^g}\{g, N\}$, where N is the nonce, in place of just g. Figure 4 shows the message format for this scheme.

Nonce	MAC of group identifier	Encrypted Payload	MAC over entire message
N	$H_{k_v^g}\{g,N\}$	$\{payload\}_{k_v^g}$	$H_{k_v^g}\{N, H_{k_v^g}\{g,N\}, \{payload\}_{k_v^g}\}$

Figure 4: Traffic Analysis resistant message construction

5.4 Security Analysis

Using the message of Figure 1, anybody with access to the network packets could know which group they belong to. With the new message of Figure 4, this cannot be done. Based only on the group protocol fields, two different messages of the same group cannot be correlated. The reason being that all fields in the two messages would be different and completely unrelated, making the messages seem completely independent to somebody who's not a group member. However, as the same message goes to each member, if an adversary knows somehow that two people accepted the same message (such as by timing any replies), it could conclude

that they belong to the same group. Thus, the new message construction restricts traffic analysis to a per-message level: an adversary has to observe the same message at different points in the network to establish anything meaningful from it.

5.5 Performance

There are several performance questions we wish to answer for the above scheme. Firstly, what is the overhead of using the new message construction in the setting of just one group per member. Next, what is the cost of cycling through the <group identifier, group key> list when there are more than one group per member. Another metric is the time required to discard a potentially malicious message. We also want to know what is the maximum throughput the scheme can support.

For measuring the performance numbers, we wrote a program that generates and processed messages. Generating a message requires just putting in the various fields according to the message format. For the payload, we use a constant string whose size can be specified as a parameter. Processing a message for the simple format of Figure 1 requires just looking up the group and view identifiers, using them as index to get the group key, verifying the MAC over the entire message and decrypting the payload. Processing a message of the traffic analysis resistant format of Figure 4 requires cycling through the <group identifier, group key> list to match the message with a group, verifying the MAC over the entire message and decrypting the payload.

There are several parameters to our experiments: number of groups per member (size of the list through which to cycle), payload size, encryption and MAC algorithms. We carried out several experiments varying all these parameters but for reasons of space cannot present all the results here. However, all our results follow similar patterns and thus presenting only the important results should suffice. For the experiments presented in this paper, we chose to vary the number of groups per member and the payload size while fixing the crypto algorithms. We used `Blowfish` with 160 bit keys for encryption and `MD5` based HMAC for the MAC. Further, we used 8 byte long group identifiers, 4 byte long view identifiers and 4 byte nonces.

The machine on which experiments were run on was a IBM Netfinitity with a 1 GHz PIII processor, 256 MB of RAM and running the Linux operating system. Our program was written in C++ and used the `openssl` library for all crypto functionality. We note that because of the CPU intensive nature of our scheme, the actual time and throughput figures in our paper are extremely sensitive to the CPU speed. It would be more or less correct to say that given a processor that is 2 times faster, the times would be halved.

Figure 5 presents two performance graphs that are representative of the graphs we got. The graphs compare the performance of the message formats of Figure 1 and Figure 4. The throughput was calculated as:

$$throughput\ (in\ MBps) = \frac{payload\ size\ (in\ bytes)}{message\ processing\ time\ (in\ sec)}$$

For all our experiments, we used a uniformly random distribution of messages to groups. Thus, for the traffic analysis resistant scheme, on average half of the group list is traversed before getting a match. It can be seen from Figure 5 that the time required to process a traffic analysis resistant message increases linearly with the number of groups per member, whereas the time to process a simple message is independent of number of groups per member. This is indeed what we would expected.

Figure 6 shows the relative cost of processing a traffic analysis resistant message compared to the cost of processing a simple message. Remember that processing a traffic analysis resistant message involves cycling through the group list while computing MACs, verifying the MAC over the entire message and decrypting the payload. On the other hand, the simple message only requires verifying MAC over entire message and decrypting the payload The cost to traverse the <group identifier, group key> list is independent of payload size, whereas the costs for decrypting the payload and verifying MAC over the entire message increase with payload size. Therefore, the relative overhead of cycling through the group list gets smaller as the payload

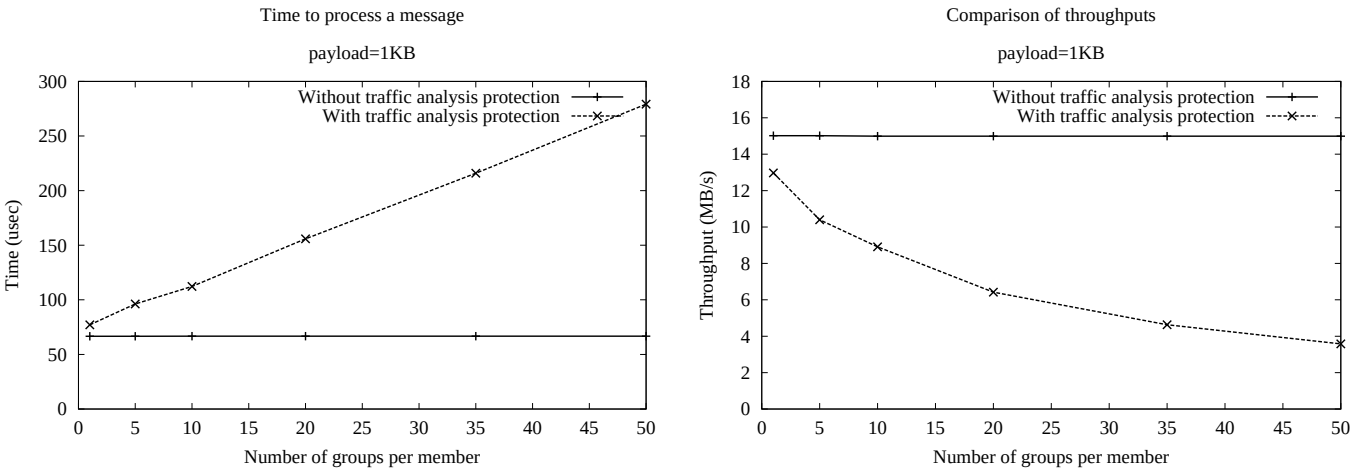

Figure 5: Performance of traffic analysis prevention: Typical graph

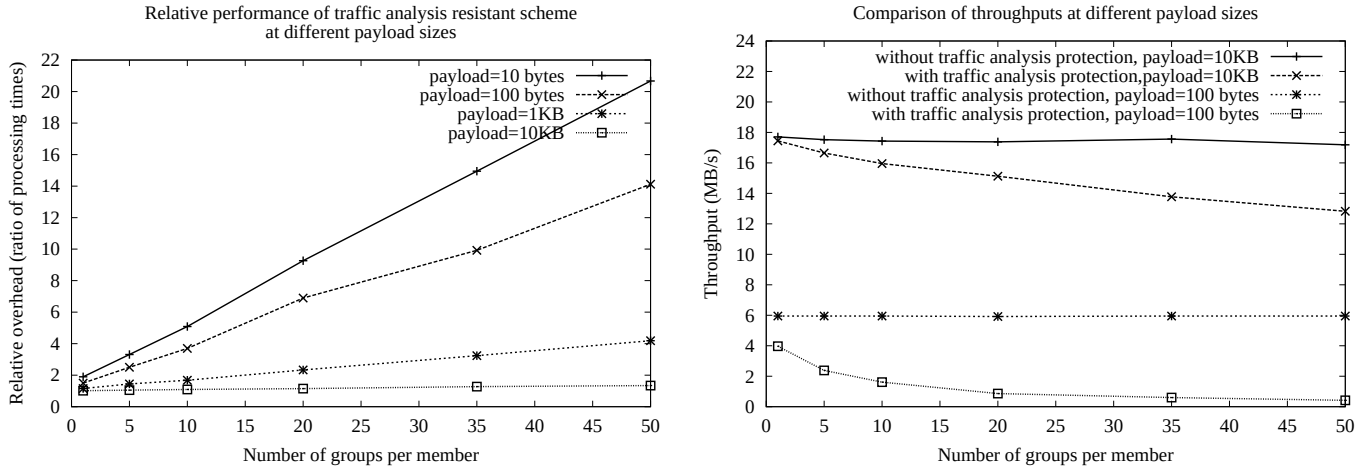

Figure 6: Performance of traffic analysis prevention: Effect of different payload sizes

gets larger. This can be seen from the graphs. The traffic analysis resistant scheme performs pretty well when the payload is large (10 KB), always staying within a factor of 2 of the sample message format. However, as the payload gets smaller, the performance of the traffic analysis resistant scheme degrades rapidly as the number of groups per member increases.

Figure 7 shows the cost of discarding a potentially malicious message. This is the worst case for the traffic analysis scheme as the group list needs to be traversed completely. To discard a message, the sample message scheme of Figure 1 just needs to verify the MAC over the packet which would fail. On the other hand, the traffic analysis resistant scheme of Figure 4 needs to iterate through the complete list and also verify the MAC over the entire message which would fail. The group list needs to be traverses completely because we assume that the receiver finds a match at the last entry. We note that an adversary can cut and paste the group identifier field and in the worst case, this would make a receiver find a match at the last entry. The MAC over the entire message would still fail though. The minimum attack bandwidth was

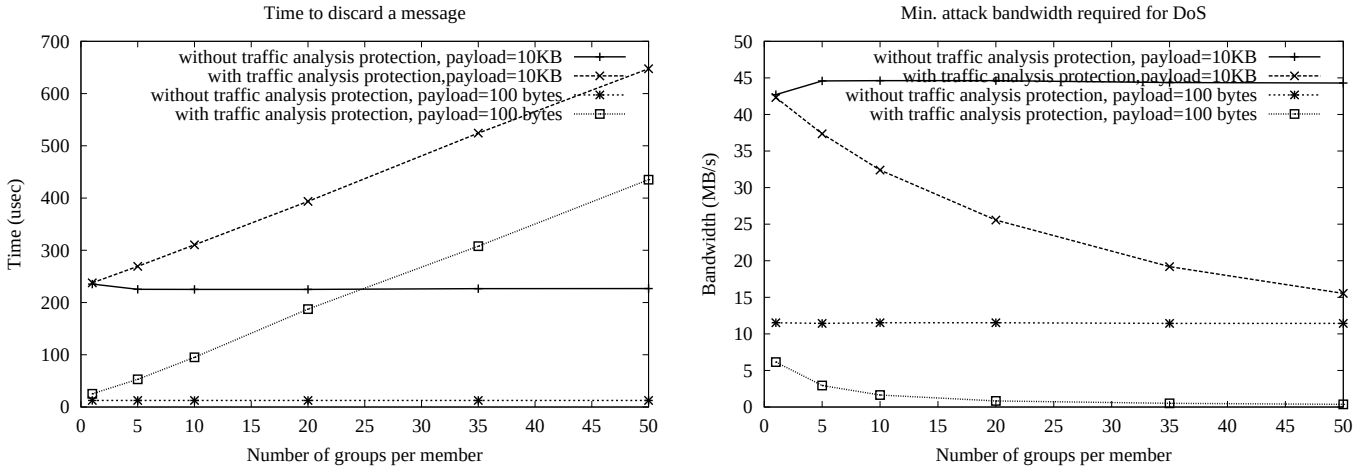

Figure 7: Performance of traffic analysis prevention: Discarding messages

calculated as

$$min.\ bandwidth\ (in\ MBps) = \frac{total\ packet\ size\ (in\ bytes)}{message\ processing\ time\ (in\ sec)}$$

As seen from Figure 7, if the payload is small, our scheme is vulnerable to computational denial of service attacks requiring only a small attack bandwidth.

Message Size: Another performance parameter for our scheme is the increase in message size. With the group identifier and nonce sizes we used (8 bytes and 4 bytes respectively) and the crypto algorithms we used, the size of the traffic analysis resistant message was: 20 bytes + payload size + size of MAC over entire message. For the sample message, the size was 8 bytes + payload size + size of MAC over entire message. Thus, the traffic resistant message was 12 bytes longer than the sample message. This would be a concern only if the payload was very small, of the range of tens of bytes. We believe that such a small payload is unrealistic in real-world applications and thus message size is not a reason for concern.

5.6 Optimizations

In this section, we describe three optimizations that could significantly improving the performance of our scheme. Here, we only sketch out the ideas of these three schemes leaving out the details. However, based on our performance results in the previous section, we believe that these optimizations can significantly improve the performance.

Intelligently ordering the group list

This is probably the first thing that would come to anybody's mind. The idea is to use traffic patterns to better organize the <group identifier, group key> list so that active groups (which are sending packets frequently) are placed near to the top with less active groups nearer to the bottom of the list. Doing so would decrease the average time needed to match a message (that is to be accepted) with its group. One simple way to organize the list is to move group g to the head of the list whenever a packet for it is received. Obviously, other more ingenious schemes can be imagined. We note that in the worst case (packet is to be discarded), this scheme does not help as the whole list needs to be traversed.

Splitting the address/search space

A fundamental limitation of our scheme is that, in the worst case, the whole list needs to be traversed. This linear relationship scales poorly when the receiver is member of a large number of groups. We can get rid of this problem by splitting the search space (or the address/name space for the group identifier). For example, we could split the group identifier field into two sub-names. The message then would contain two MAC fields (one for each part of the group identifier) instead of just one. There will also be two keys to compute those two MACs field[3]. Each user will maintain two lists, one for each part of the group identifier. Let's say a user is member of 256 groups and the group identifiers are 8 bits each. Thus, the list the user needs to maintain contains 256 entries, meaning that there need to be a maximum of 256 MAC checks. Now, let's suppose that the group identifier is split into two halves of 4 bits each. Two lists will be maintained, each containing a maximum of $2^4 = 16$ entries. Thus, the maximum work needed will be only $2 * 16 = 32$ MAC checks.

Using the splitting technique weakens the unlinkability properties of our scheme. The reason is that groups sharing a sub-name have to share the key corresponding to that sub-name. Thus, a common sub-name is revealed to all groups that share that sub-name. For example, consider two group identifiers: *abcd* and *abef* which are split into (ab, cd) and (ab, ef) respectively. If a packet is sent for *abcd*, *abef* can know that the packet is for some group named $abXX$, where the X's are unknown.

Another limitation of this scheme is that because keys need to be shared between different groups, it can only be employed in situations where the groups are cooperatively managed. However, when it can be employed and when the weakened security guarantee is acceptable, this scheme can dramatically improve performance. We also note that the scheme is very flexible. The group identifiers need not be split into halves as we have discussed but can be split in any way. This permits various security-efficiency trade-offs.

Pre-distribution of Nonces

The idea in this scheme is for senders of group G to pre-distribute the nonces to be used in future messages it will send. Members of G can pre-calculate the expected MAC field. Whenever they receive a packet with one of the pre-distributed nonces, they only have to compare the pre-calculated MAC with the MAC field in the packet. If these two match, there is no need to iterate through the <group identifier, group key> list. On the other hand, if there is not a match, the receiver just follows the normal procedure of iterating through the list.

Nonces can be efficiently pre-distributed by piggybacking them on other group messages. Membership changes pose some problems to this scheme as members who left the group could use the pre-distributed nonces to figure out that a message has been sent for the group. Thus, senders should discard their pre-distributed nonces whenever a membership change occurs. As with the list ordering scheme, this scheme can potentially improve the average performance but not the worst case performance.

5.7 Traffic Analysis in Ethernets

As Ethernets are a broadcast medium[4], all traffic sent out by any host on an Ethernet is readable by all other hosts on the same Ethernet. Thus, anybody on the Ethernet can do the following:

1. know *what* traffic is going on the Ethernet
2. know *for whom* the traffic is destined or *from whom* it was sent

[3]although, we used the group key to compute the MAC of the group identifier, we could have used some other key too that is known to all group member. So, the key with which MAC is taken is independent of the group key.

[4]we omit switched Ethernets from our discussion.

Thus, an Ethernet offers absolutely no security from other hosts on the Ethernet unless security is implemented at a higher layer. Even if this is the case, there is no protection from traffic analysis based on the MAC/network/transport headers and thus the second problem above still remains. Moreover, since Ethernets present a broadcast medium, the model of the adversary is one that can monitor *all* packets on the network. Thus, anonymity solutions discussed in Section 4 either completely fail (Crowds, Hordes) or become more vulnerable (Mix-net, Onion routing) as pointed out in [35]. Moreover, these solutions turn out to be very expensive both in terms of latency as well as throughput. For example, consider the situation that a packet is routed via 4 other machines to disguise the source and destination of the packet. Then, for a x bytes packet, approximately $4x$ bytes will be put on the wire, reducing the throughput by a factor of 4. Latency would also go up by a similar factor.

We can model the communication on an Ethernet as group communication and apply the techniques from Section 5.3. Let us assume for simplicity that messages on an Ethernet are either two-party messages or are broadcast to all hosts on the Ethernet. An Ethernet also has a gateway machine for sending/receiving messages to machines outside the Ethernet. Our goal here is to protect the communication only within the Ethernet (and not outside it) from traffic analysis.

The communication in the Ethernet is mapped to groups as follows. Each pair of hosts on the Ethernet form a two-party group. There is one more group for broadcast and it consists of all hosts on the Ethernet. Thus, each pair of hosts (I, J) share a symmetric key k_{IJ} and there is one more key for the broadcast group. Thus, on an Ethernet with p hosts (including the gateway), there are $\binom{p}{2} = \frac{p(p-1)}{2}$ two-party groups and 1 broadcast group. In all, there are p groups per host.

Group keys associated with the groups as created above can now be used to send traffic-analysis resistant messages on the Ethernet. This should be done at the lowest-possible layer in the protocol stack (preferably replacing the MAC layer so that MAC addresses also cannot be used for traffic analysis). For a host I to send a message to host J, the message header can be: $N, H_{k_{IJ}}\{(I, J), N\}$ (we use (I, J) to name the group containing I and J). On receiving a message, each host would cycle through its list of p keys and see if any key generates the same MAC value. If it can, the message is further processed, otherwise it is discarded.

For packets exchanged between a host and the network beyond the Ethernet, the above protocol would be used to send messages in a traffic-resistant manner up to the gateway. The gateway can then deploy an appropriate protocol for sending messages subsequently (e.g., IPSec) or a version of our protocol.

One important question is whether the performance is likely to be adequate to saturate the Ethernet. Clearly, the value of p, the number of hosts on the Ethernet, is an important number. For small Ethernets (e.g., $p = 10$), using the data in Figure 5, it appears that we can achieve approximately 8 MB/sec throughput (assuming 1 KB messages) with software-based hashing on Pentium III machines. In practice, however, the above computation would have to be implemented on the Ethernet card in hardware, so that the CPU on the machine is free to do other work. Memory requirements are not significant because there are only a few (16-20) bytes per key. Assuming hardware performance that is 10-times faster than we got with a software implementation, it appears that our scheme will be able to sustain throughputs of 100 Mb/sec to 1Gb/sec on an Ethernet.

For a large Ethernet such as a class C LAN, the worst case will occur when there are the maximum of 256 nodes. A cheap hardware-based crypto Ethernet card appears adequate to achieve throughput for 10 Mb/sec Ethernets at current speeds. For higher throughputs, we can use the space-splitting approach suggested in Section 5.6. We could have two lists of 16 entries each instead of one big list of 256 entries. This would reduce the maximum work required at each machine to 32 MAC checks, down from 256 MAC checks, while still providing significant protection against traffic analysis. This would improve expected throughputs to about 100 Mb/sec range with common hardware technology and possibly higher if hash computations and comparisons can be done in parallel for all the keys. With throughputs of hardware crypto cards going up all the time as hardware gets faster, it appears to be within realm of possibility to make Ethernet traffic resistant to traffic analysis even at very high throughputs and for large Ethernets.

6 DoS and Traffic Analysis resistant key recovery protocol

In this section, we present our solution to the problem of secure key recovery in group protocols. We first outline our goals, then present our solution, analyze its security and performance costs.

6.1 Goals

The key recovery procedure involves two steps: 1) a receiver who missed a rekey infers from the group packets that it has missed a rekey event, and 2) it requests the latest key from the key distributor. We note that the problem with key recovery lies in the inference step. The key request step is straightforward: the key distributor could follow a protocol similar to the rekey protocol to hand the latest key to the requester. As mentioned briefly in Section 3.2, there are two goals for a secure inference procedure:
authentication: the inference step should be authenticated. That is, a member who missed the rekey event should be able to verify that the group has indeed rekeyed. Otherwise, the protocol is vulnerable to DoS attacks as described in Section 3.2.
inference confidentiality: the inference step should be confidential, i.e. limited only to group members. More formally, whenever the view changes[5] from v_i to v_{i+1}, only group members in view v_i should be able to infer that the view has changed.

6.2 Solution

Authenticity of the inference step can be achieved by employing a commonly used technique in security called hash-chaining (as discussed in [15, 12, 28, 23]). The idea is that the view identifiers (maintained at the key distributor) are linked in the form of a hash chain. That is, the relation $h(v_{i+1}) = v_i$ holds for all i, where h is a secure hash function such as `MD5` or `SHA1`. The last hash value (v_0) is the first one to be revealed and is securely distributed to all group members using, for example, asymmetric encryption. Whenever a rekey occurs (say from view $v_i \rightarrow v_{i+1}$), the key distributor sends the new view identifier (v_{i+1}) along with the new group key. Authenticity is provided by the fact that only the key distributor can release view identifiers that are consistent with the hash relation. All subsequent group messages will now use the new view identifier, v_{i+1}. When a member who missed the rekey gets such a message, it just authenticates it by checking if the relation $h(v_{i+1}) = v_i$ holds.

Achieving confidentiality of the inference step is not as straightforward though. The reason being that the scheme has to assume that the receiver of the message might not have the latest group key (because that is the function of making such an inference). Thus, there remains no secret to work with to achieve confidentiality. Therefore, we need to introduce another key called the *view key* for this purpose. The view key is known only to group members and is used to encrypt the view identifier.

In Section 6.2.1, we present a straightforward solution using this idea: it maintains the view identifiers as a hash chain and encrypts the view identifier with the view key. We then analyze the security it provides and show that the it has a problem. This motivates the second solution (Section 6.2.2) which does not suffer from the drawback of the first one.

While describing the messages, we do not show the key distribution/agreement protocol and associated messages. We just assume that such a protocol exists and guarantees secure key distribution. We also assume that the key distribution messages themselves do not let non-members know of a rekey. The messages we show are just the group messages *from which* the inference step is performed.

Nonce	MAC of group identifier	Encrypted View Identifier	Encrypted Payload	MAC over entire message
N	$H_{k^g_{v_i}}\{g, N\}$	$\{v_i, N\}_{vk_i}$	$\{payload\}_{k^g_{v_i}}$	$H_{k^g_{v_i}}\{N, H_{k^g_{v_i}}\{g, N\}, \{v_i, N\}_{vk_i}, \{payload\}_{k^g_{v_i}}\}$

Figure 8: Group message for key recovery solution 1

6.2.1 Solution 1

Figure 8 presents the group message for this solution. The views form a hash chain so that the relation $h(v_{i+1}) = v_i$ holds. vk_i is the view key used to encrypt v_i and is distributed to all group members during the view transition $v_{i-2} \rightarrow v_{i-1}$. Remember that users maintain a list of <group identifier, group key> pairs (Section 2.3). Now, in addition to that list, they will also maintain a list of <group identifier, view key> pairs where the view key stands for the latest view key for that group.

Let's say that a group member missed the rekey message corresponding to the view change $v_{i-1} \rightarrow v_i$. The member would still have the view key vk_i though, because it was distributed during the previous rekey (for the view change $v_{i-2} \rightarrow v_{i-1}$). The view field in messages of the new view v_i would contain the value $\{v_i, N\}_{vk_i}$ in the view field. When the member gets such a packet, it performs the following steps:

1. Cycle through the <group identifier, group key> list to match the packet with a group. This step would fail because the current group key is not known. This is just the procedure followed for our traffic analysis scheme of Section 5. It is not the inference step.
2. Cycle through the <group identifier, view key> list and repeat the steps below until it is found which group the message is for, or until all groups have been tried without success meaning that the packet is to be discarded:
   (a) use the group's view key to decrypt the view field and get the candidate for the new view. Let's call it v_i'.
   (b) check if $h(v_i') = v_{i-1}$. If so, accept the new view and request new key.

Security Analysis

Authenticity: We assume that the new view identifier is revealed along with the new group key. This coupled with linking the view identifiers in a hash chain provides authenticity of the inference step.

Inference confidentiality: The view is encrypted using the view key, guaranteeing that only current group members can read the view identifier. We note that once again the nonce plays an important role by ensuring freshness of the view field across messages of the same view. If the nonce was not present, then the view field would be the same for all messages within a view and by monitoring changes in the field, an adversary can know when the view changes.

There is still a slight problem with the confidentiality provided though. As all the view identifiers form a single hash chain, any two of them can be linked with each other. Using this fact, anybody who was a group member in view v_a, then left the group and joined again in view v_b $(a < b)$ would be able to know exactly how many view changes the group has gone through. This is not desirable and the next solution we present does not contain this flaw.

Nonce	MAC of group identifier	Encrypted View Identifier	Encrypted Payload	MAC over entire message
N	$H_{k^g_{v_i}}\{g, N\}$	$\{v^1_i, N\}_{v^0_i}$	$\{payload\}_{k^g_{v_i}}$	$H_{k^g_{v_i}}\{N, H_{k^g_{v_i}}\{g, N\}, \{v^1_i, N\}_{v^0_i}, \{payload\}_{k^g_{v_i}}\}$

Figure 9: Group message for key recovery solution 2

6.2.2 Solution 2

Figure 9 presents the group message for this solution. Now the views *do not* form a hash chain. Instead, each view identifier is split into two parts which are related by a hash relation. The two parts of view v_i[6] are v_i^0 and v_i^1, and they are related by the relation $h(v_i^1) = v_i^0$. Neither v_i^0 nor v_i^1 is related in any way to preceding or succeeding view identifiers. v_i^0 is used as the view key and the hash relation between v_i^0 and v_i^1 serves as the authentication mechanism. As with solution 1, the view key (i.e. v_i^0) is distributed during the transition $v_{i-2} \rightarrow v_{i-1}$. v_i^1 is revealed only when the view changes. Thus, it is distributed along with the transition $v_{i-1} \rightarrow v_i$.

Let's say that a group member did not get the key for view v_i. The member would still have v_i^0 because it was distributed during the previous rekey. Packets in the new view v_i would have the field $\{v_i^1, N\}_{v_i^0}$. The protocol steps to be followed with this solution are exactly the same as for solution 1 except for the following changes:

1. v_i^0 is the view key now
2. the hash relation to check is: $h(v_i^1) = v_i^0$

Security Analysis

Authenticity: This solution also uses a hash relation similar to the one used for solution 1 and provides authenticity for the same reasons.

Inference confidentiality: As for solution 1, a view key is used to encrypt the view identifier so that only current members can read it. The improvement over solution 1 is that the view identifiers for different views are completely independent. Thus, the flaw that existed in solution 1 is not present any longer.

6.2.3 Multiple Key Losses: A discussion

We note that both the schemes presented above provide a mechanism to recover from only a single key loss. If there are multiple key losses (for example, from a bursty message loss), the group member would not have the appropriate view identifiers to decrypt the view field and check the hash relation.

It is simple to extend our schemes to handle m key losses, where m is a fixed quantity. The idea is to pre-distribute m view keys (for the next m membership changes) rather than just one. Any member in view v_a would have the view keys v_{a+j}^0, where $1 \leq j \leq n$. When a message is received, all m view keys will be tried and the corresponding hash check performed. This will allow members to detect the loss of j rekey messages where $1 \leq j \leq m$. For example, let's say a group g is in view v_x and goes to view v_{x+1}, then to v_{x+2} and then to v_{x+3}. That is, it goes through a sequence of 3 view changes. Suppose that a member lost all the 3 rekey messages. For the scheme to work, $m \geq 3$ and we assume this. Therefore, the member would still have v_{x+3}^0. Sometime during the phase when it is checking the view identifier field it would try v_{x+3}^0 as

[5] remember that view change and rekey go hand in hand

[6] although there is no identifier v_i in this scheme, we continue to use the notation for convenience in writing and better readability. That is, v_i only means i^{th} view.

the view key and get v^1_{x+3} after decryption. Then it would check if $h(v^1_{x+3}) = v^0_{x+3}$ which would be true. This would tell the member that the group g has changed view 3 times.

We note that this extension has the following 2 consequences:

1. the confidentiality guarantee is weakened. Anybody who was a group member in view v_x can know that the view has changed to v_{x+j}, $j \leq m$, even if it is not a group member some of those views.
2. the time to discard a message increases proportional to m, as all m view keys need to be tried for all groups someone is member of.

6.3 Performance

We note that the view identifier field would be checked in only two situations: 1) member has lost a rekey, and 2) when discarding a message (before discarding, it needs to be ensured that the packet does not belong to a group whose rekey event was missed). During normal operation of the group protocol (that is, when processing valid packets), only the <group identifier, group key> list would be traversed to match the packet with its group; the view field would just be ignored and the performance would be as shown in Figure 6.

The worst case for our scheme is when the message needs to be discarded (case 2 above). In that case both the <group identifier, group key> and <group identifier, view key> lists will be traversed completely. Here, we present results for only this case.

The setup we used for obtaining performance results is the same as the one used and described in Section 5.5. We compare three protocols:

- Traffic analysis resistant protocol that prevents DoS attacks on key recovery phase as well as provides inference confidentiality, as presented in Solution 2 (Figure 9).
- Traffic analysis resistant protocol that does not prevent DoS attacks on key recovery phase, as presented in Figure 4.
- Simple protocol that is not resistant to traffic analysis and does not prevent DoS attack on key recovery phase, as presented in Figure 1.

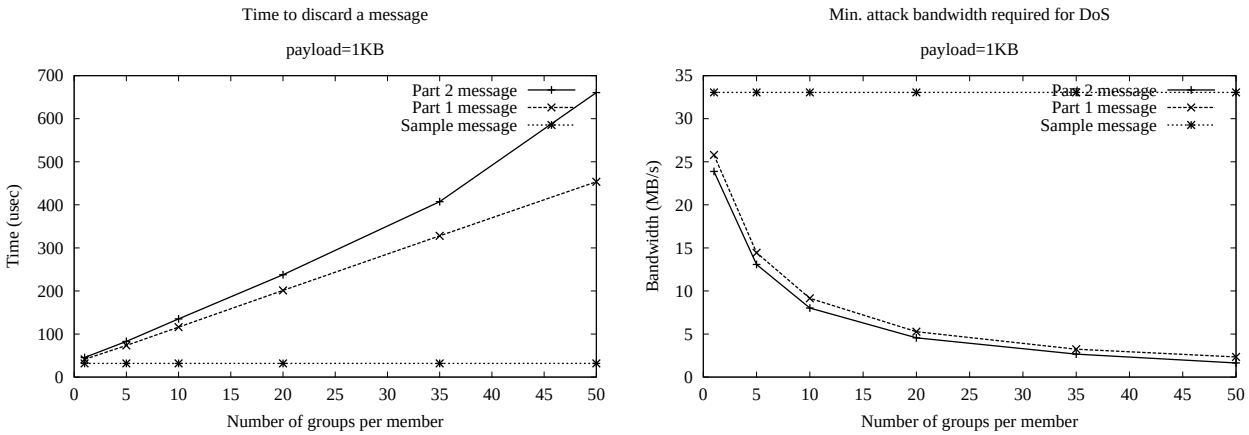

Figure 10: Performance of secure key recovery: Discarding messages

We used a 20 byte v^1_i field and a 4 byte nonce. As in Section 5.5, `Blowfish` and `MD5` based HMAC were used for encryption and MAC respectively. MD5 was also used as the secure hash function. The length of

the v_i^0 field is equal to the MD5 output, that is, 16 bytes. Figure 10 presents the time required and the associated maximum throughput when discarding a message. We make the following observations:

- If the number of groups/member is small (close to 1), the overheads are negligible, even for a scheme that provides both traffic analysis protection and inference confidentiality.
- If the number of groups/member is large, providing both traffic analysis protection and inference confidentiality becomes expensive. Either hardware accelerators for crypto operations are likely to be required or the scheme would be applicable only to situations requiring low-throughput (e.g., military environments over slow satellite links).

If only authentication is required, just hash-chaining the view identifiers would suffice. This would be similar to schemes used in Tesla and other hash-chain based signature schemes. For m key losses to be tolerated, the scheme would require up to m hash computations (not MACs) on part of a member, irrespective of the number of groups/member. This would be able to provide high throughputs in filtering out malicious messages. However, the scheme would not provide protection against traffic analysis or provide inference confidentiality regarding membership changes in a group.

Message size: The sample message of Figure 1 used a 4 byte view field. With the view and nonce sizes and crypto algorithms that we used for our performance results above, the size of the view field is 32 bytes. That means an additional 24 bytes per message. Because the view field might be checked only rarely, this increase in size is probably more bothersome than that for the traffic analysis message. However, it is still a negligible increase except for very small payloads.

7 Conclusion and Future Work

In this paper, we have studied the issue of anonymity and privacy of a group. We have presented modifications to secure group communication message formats to make them resistant to traffic analysis at the group protocol level. The scheme is applicable to both data messages and key distribution messages. We also presented a scheme that allows group members to securely infer from group messages that they have missed a rekey event. The scheme provides *inference confidentiality* – non-members cannot infer that the group has changed keys, only members can.

We presented several performance results for our schemes and discussed optimizations such as address space splitting and the use of hardware accelerators to achieve sufficient performance. Based on our results, we think that performance is going to be adequate for many applications, including preventing traffic analysis on Ethernets.

Several open issues remain for further investigation. This paper assumed the use of a broadcast medium to provide aggregation of traffic. This property is crucial to achieving anonymity of a group. The issue of whether such broadcast mediums are feasible from a performance point of view and how they can be built needs to be further studied.

We would also like to explore ways of using anonymity solutions such as Crowds and Onion Routing in the setting of group communication, that is, when the traffic is multicast rather than unicast. There might be ways to use our techniques in conjunction with these protocols that leads to better solutions.

Another issue that we would like to explore is the use of our key recovery solution in systems that use key agreement protocols for generating new group keys – at this point, we have primarily focused on systems that rely on key distribution protocols for rekeying a group.

We also want to investigate if performance optimizations to our schemes are possible that lower the costs of rejecting maliciously injected messages, while continuing to provide similar anonymity guarantees.

Currently, we are implementing our solution for providing anonymity in Ethernets, along with all the suggested optimizations. This will provide real, experimental performance data and also serve as a test-bed for any future work.

Acknowledgments

We would like to thank Jim Irrer for bringing the problem of secure key recovery to our notice and for useful discussions and insightful comments on the issue of anonymity in group communication.

References

[1] Y. Amir, G. Ateniese, D. Hasse, Y. Kim, C. Nita-Rotaru, T. Schlossnagle, J. Schultz, J. Stanton, and G. Tsudik. Secure Group Communication in Asynchronous Networks with Failures: Integration and Experiments.

[2] Secure Spread webpage. http://www.cnds.jhu.edu/research/group/secure_spread/.

[3] The Anonymizer. http://www.anonymizer.net/.

[4] M. Bellare, R. Canetti, and H. Krawczyk. HMAC: Keyed hashing for message authentication. Internet Request for Comments, RFC 2104, Internet Engineering Task Force, Feb 1997.

[5] M. Burrows, M. Abadi, and R.M. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8, February 1990.

[6] D. Chaum. Untraceable electronic mail, return addresses, and digital pseduonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[7] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[8] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[9] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private Internet connections. *Communications of the ACM (USA)*, 42(2):39–41, 1999.

[10] Onion Routing webpage. http://www.onion-router.net/.

[11] Y. Guan, C. Li, D. Xuan, R. Bettati, and W. Zhao. Preventing Traffic Analysis for Real-Time Communication Networks. In *Proceedings of The IEEE Military Communication Conference (MILCOM)*, 1999.

[12] N.M Haller. The S/Key One-Time Password System. In *Proceedings of 1994 Internet Society Symposium on Network and Distributed Systems Security*, pages 151–157, San Diego CA, USA, February 1994.

[13] IPSec webpage. http://www.ietf.org/html.charters/ipsec-charter.html.

[14] S. Jiang, N. Vaidya, and W. Zhao. Routing in packet radio networks to prevent traffic analysis. In *Proceedings of the IEEE Information Assurance and Security Workshop, West Point, NY*, June 2000.

[15] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11), Nov. 1981.

[16] P. McDaniel and A. Prakash. Antigone: Implementing policy in secure group communication. Technical Report CSE-TR-426-00, Electrical Engineering and Computer Science, University of Michigan, 2000.

[17] P. McDaniel, A. Prakash, and P. Honeyman. Antigone: A Flexible Framework for Secure Group Communication. In *Proceedings of the 8th USENIX Security Symposium*, pages 99–114, August 1999.

[18] R.C. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, April 1980.

[19] R.M. Needham and M.D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[20] B. C. Neuman and T. Ts'o. Kerberos: An authentication Service for Computer Networks. *IEEE Communications*, 32(9):33–38, September 1994.

[21] A. Perrig. The BiBa One-Time Signature and Broadcast Authentication Protocol. In *Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8)*, pages 28–37, Philadelphia PA, USA, November 2001.

[22] A. Perrig, R. Canetti, B. Briscoe, J. Tygar, and D.X. Song. TESLA: Multicast Source Authentication Transform. Internet Draft, Internet Engineering Task Force, July 2000.

[23] A. Perrig, R. Canetti, J. Tygar, and D.X. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, May 2000.

[24] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[25] R.L. Rivest. The MD5 message-digest algorithm. Internet Request for Comments, RFC 1321, April 1992.

[26] R.L. Rivest, A. Shamir, and L.M. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[27] O. Rodeh, K. Birman, M. Hayden, Z. Xiao, and D. Dolev. Ensemble Security. Technical Report TR98-1703, Cornell University, September 1998.

[28] Aviel D. Rubin. Independent One-Time Passwords. *USENIX Journal of Computer Systems*, 9(1):15–27, February 1996.

[29] B. Schneier. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). In *Cambridge Security Workshop Proceedings*, pages 191–204, 1994.

[30] Clay Shields and Brian Neil Levine. A Protocol for Anonymous Communication Over the Internet. In *Proceedings of the* 7^{th} *ACM Conference on Computer and Communication Security*, 2000.

[31] J. G. Steiner, B. C. Neuman, and J. J. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the Usenix Conference*, pages 191–202, 1988.

[32] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A New Approach to Group Key Agreement. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, pages 380–387, Amsterdam, 1998. IEEE Computer Society Press.

[33] Debby M. Wallner, Eric J. Harder, and Ryan C. Agee. Key Management for Multicast: Issues and Architectures. *Internet Engineering Task Force*, June 1999. RFC 2627.

[34] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communication Using Key Graphs. In *Proceedings of ACM SIGCOMM '98*, pages 68–79. ACM, September 1998.

[35] M. Wright, M. Adler, B. N. Levine, and C Shields. An Analysis of the Degradation of Anonymous Protocols. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS 2002)*, February 2002.

[36] Y. G. Xinwen. Efficient Traffic Camouflaging in Mission-Critical QoS-guaranteed Networks. In *Proceedings of the IEEE Information Assurance and Security Workshop, West Point, NY*, June 2000.