M2: Multicasting Mixes for Efficient and Anonymous Communication

Ginger Perng Michael K. Reiter Chenxi Wang
Carnegie Mellon University
{gperng,reiter,chenxi}@cmu.edu

Abstract

We present a technique to achieve anonymous multicasting in mix networks to deliver content from producers to consumers. Employing multicast allows content producers to send (and mixes to forward) information to multiple consumers without repeating work for each individual consumer. In our approach, consumers register interest for content by creating paths in the mix network to the content's producers. When possible, these paths are merged in the network so that paths destined for the same producer share a common path suffix to the producer. When a producer sends content, the content travels this common suffix toward its consumers (in the reverse direction) and "branches" into multiple messages when necessary. We detail the design of this technique and then analyze the unlinkability of our approach against a global, passive adversary who controls both the producer and some mixes. We show that there is a subtle degradation of unlinkability that arises from multicast. We discuss techniques to tune our design to mitigate this degradation while retaining the benefits of multicast.

1. Introduction

A mix [5] is a routing element that attempts to hide the correspondences between its input and output messages, i.e., so an observer cannot determine which output message corresponds to a particular message that the mix received. To achieve this, a mix typically transforms each message it receives (e.g., by decrypting it) and then outputs messages in an order different from that in which it received them. If the compromise of a single mix is feared, then a message can be routed through multiple mixes (a *mix network*) to hide the correspondence between the message originator and destination (provided that at least one mix remains uncompromised), a property called *unlinkability* [13].

In this paper we focus on the response to a message routed through a mix network. The response should be routed through the mix network so as to not disclose the correspondence between the originator and responder, and without requiring the responder to know the identity of the originator (so the originator can remain anonymous to the responder, if desired). An example of a system with such properties is a Type-II anonymous remailer (e.g., [11]), at which a user anonymously registers an email account by routing a registration message through a mix network. In doing so, it deposits at the server a data structure that enables the server to route an email back to the (still anonymous) user along the registration path in the reverse direction. As this model of registering for content to be returned in the future is our primary motivation, we henceforth refer to the originator as the content *consumer* and the responder as the content *producer*.

In a scenario where there are multiple consumers to which the content should be sent (e.g., an email to a mailing list), an anonymous unicast network would have the producer send the content to each consumer individually, incurring costs at the producer and the mixes that are linear in the number of consumers. In this paper, we present a protocol called M2 that permits efficient anonymous communication through a multicast-like mechanism. In our approach, mutually-trusting consumers can autonomously form a *consumer group* (or just *group*). Each consumer can register individually with the content producer by routing a registration through the mix network, though a mix on this path might "merge" this registration into a previous registration by a member of the same group, thus forwarding it no further. When the mix receives content traveling on the first registration path (but in the reverse direction), the mix duplicates this content along each path that the mix merged into this original path. In this way, our technique disseminates content to a consumer group on a multicast "tree" formed by merging registration (and hence content distribution) paths. In particular, registrations from members of the group can appear to the producer as a single registration.

For example, Figure 1 illustrates the paths via which consumers c_0, c_1, and c_2 in the same group register with a producer r and the reverse of which is used by the producer to route content to these consumers. Notice that c_0, c_1 and c_2 share mix_0 as the third mix on their subscription paths,

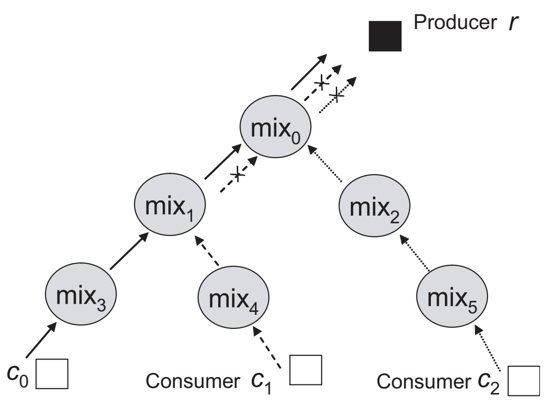


Figure 1. Merging registration paths.

and that c_0 and c_1 share mix_1. In this case, our protocol enables r to send only a single content message to mix_0, with the same computation complexity as if there were only one consumer. Similarly, mix_0 forwards a content message, after appropriate processing, to each of mix_1 and mix_2; the needed processing is virtually the same as if mix_0 had received separate messages destined for mix_1 and mix_2. In this way, our protocol achieves multicast-like dissemination of content, while still permitting mixing.

In this paper we detail M2's algorithms to achieve anonymous multicast and analyze its properties in a model permitting a global, passive adversary that controls the producer and some mixes. We show that enabling multicast within mix networks leads to a subtle and intrinsic degradation of anonymity. This degradation arises from a mix performing a "branch" for a multicast, e.g., when mix_0 forwards the content message to both mix_1 and mix_2. A consequence of this branching is that mix_0 necessarily learns additional information about the content, namely that it is of interest to at least two consumers (in one group). Combined with *a priori* information about the possible number of consumers for producers and possible group sizes, which we conservatively permit the adversary to have, this can lead to the discovery of the likely producer of this content message and hence link a consumer to the producer of the content it receives. Perhaps surprisingly, we show that traditional mix network architectures that provide strong unicast unlinkability, notably mix cascades [12], are vulnerable to this weakness when multicast is employed.

There is thus a tension between multicast and anonymity, and so we propose a technique to strike a balance between them. We show that through careful tuning of parameters of our system, we can overcome this tension in many cases. In these cases, our techniques gain efficiency through the judicious use of multicast, while still protecting unlinkability against corrupt mixes. We quantify this tradeoff through analysis and simulation.

2. Related Work

We emphasize that our use of multicast, in a system supporting unlinkability, is different from the traditional use of multicast to achieve *receiver anonymity* [13]. Put simply, the latter use of multicast sends a unicast message to a single destination by multicasting the message to a group containing that destination. The intended destination recognizes the message as intended for itself either because it expects this message (e.g., as in Hordes [15]), or because the sender addresses the message *implicitly* [13], i.e., in a way that only the intended destination can recognize itself as the target. This use of multicast is intrinsic to the receiver anonymity guarantee and is consumptive in that it delivers messages unnecessarily: the overwhelming majority of recipients discard the message. In contrast, our goal is to implement multicast in order to *save* bandwidth over independent unicasts to the same consumers. And as we show here, it has a deleterious effect on unlinkability that must be traded off against bandwidth savings.

Current schemes to provide anonymous multicast include the Dedicated Multicast Anonymizer (DMCA) [8] and Secure and Anonymous Multicast (SAM) [16]. Similar to the Anonymizer (`http://www.anonymizer.com`) and the Lucent Personalized Web Assistant (LPWA) [7], both use a trusted proxy to hide the consumers from the producer. The trusted proxy joins the multicast tree for its consumers and forwards messages for the multicast group to the consumers. Similarly, a producer can use the proxy to hide its identity from the content consumers. However, these approaches do not implement unlinkability versus a global eavesdropper. Additionally, the proxy presents a single point of failure that, if corrupted, can eliminate the anonymity benefits of the system.

3. System Model

We assume that there is a public key infrastructure by which parties can learn the public keys (and network addresses) of mixes. In addition, mixes utilize this public key infrastructure to establish pairwise symmetric keys between them, to implement point-to-point encryption and to authenticate each other. We assume that, through a mechanism external to M2, mutually trusting consumers create groups to register interest for a particular producer's content. We emphasize that although we are not concerned with how a consumer joins a group or how groups are formed, it is the case that consumers in the same group mutually trust one another, in that the compromise of a group member can result in a loss of anonymity for the others.

One reason that this mutual trust is required is that the consumers in a group share a secret *group key* per producer; this key is used within our algorithm for routing that producer's content through mixes to the group members. We emphasize that we are concerned with protecting unlinkability against the mix network—specifically, compromised mixes, producers, and global observers—and not against an attack on the exchange of the group key. So, while we are not concerned with how this group key is shared between group members, we stipulate that it be done in a way not visible to the attackers of concern, e.g., in a face-to-face meeting or using a standard group key management protocol, e.g., [2, 4, 17], over the mix network via unicast.

As indicated above, we admit the possibility of compromised mixes. We consider these mixes to be "honest but curious," in that they follow the protocol but pool their views of the system in an attempt to break unlinkability between producers and consumers. They are also aided by producers and a global observer who views all messages sent over the network. As indicated above, we do assume that consumers within a group trust one another. Individual consumers who have similar interests join together into groups to hide their collective interest for some producer's content. By using M2, the consumers can do so with decreased network costs in comparison to retrieving the content via unicasts.

4. Multicast Path Generation

To register interest for content from a particular producer, a consumer (holding a group key) sends a registration message to the M2 network. The registration message creates a forward path through the network toward the producer of the content. Each mix along the path of the registration message processes the message, stores routing information necessary to route content in the reverse direction, and (possibly) forwards the registration to the next mix. This process is repeated until the registration message reaches the producer, thereby creating a reverse path back to the consumer. When the content is produced, a content message follows the path in the reverse direction, using the routing information established during registration to reach the consumers.

In this section, we describe the registration protocol in M2. We detail how a consumer generates a registration message in Section 4.1 and how a mix processes registration messages in Section 4.2. Lastly, we describe how a producer processes the registration message in Section 4.3.

For simplicity, we discuss the M2 protocol in the context of one consumer group. However, multiple groups can utilize this protocol simultaneously.

4.1. Registration Message

In M2, mutually trusting consumers form a group to register interest for content distributed by a producer. The group g shares a *group key* gk_g for that producer, which is used as an argument to several pseudorandom functions as described below.[1] In particular, each group member uses gk_g to create a path through the M2 network to the producer. This path is then used in reverse as a content distribution path for the producer to send content to the consumer. This path consists of a number of arbitrarily chosen mixes, $\langle \mathsf{mix}[h], \mathsf{mix}[h-1], \ldots, \mathsf{mix}[0] \rangle$, which forward the consumer's registration message toward the producer. For simplicity of presentation, we let $\mathsf{mix}[h+1]$ and $\mathsf{mix}[-1]$ denote the consumer and producer, respectively.

The registration message is a layered encryption similar to that in the original Chaum scheme [5]. For the first layer of the registration message, the consumer generates $\langle out_{\mathsf{mix}[0]}, \mathsf{ck}_g \rangle$. $out_{\mathsf{mix}[0]}$ is the "content label" which the producer labels content for $\mathsf{mix}[0]$, the last mix on the registration path, and ck_g is the content key with which the producer encrypts content for this consumer. The consumer computes these values as $out_{\mathsf{mix}[0]} = H(\mathsf{gk}_g; -1, \mathsf{mix}[0], r)$, and $\mathsf{ck}_g = G(\mathsf{gk}_g; r)$. Here, $H(\mathsf{gk}_g; \cdot)$ and $G(\mathsf{gk}_g; \cdot)$ are pseudorandom functions keyed by gk_g and for which the range is sufficiently large that the probability of a collision is negligible. The consumer encrypts $\langle out_{\mathsf{mix}[0]}, \mathsf{ck}_g \rangle$ with the producer's public encryption key (where encryption using r's public key is represented in Figure 2 by $E(r; \cdot)$).

The consumer now recursively creates encrypted layers for each mix on the message path. Starting with $\mathsf{mix}[0]$, for each mix $\mathsf{mix}[d]$ on the path, the consumer creates an encrypted layer, $z_{\mathsf{mix}[d]}$, encrypted with $\mathsf{mix}[d]$'s public key. The plaintext of $z_{\mathsf{mix}[d]}$ has the following fields:

$$\langle out_{\mathsf{mix}[d+1]}, in_{\mathsf{mix}[d]}, \mathsf{sk}_{\mathsf{mix}[d+1]}, A_{\mathsf{mix}[d]}, \mathsf{mix}[d-1], z_{\mathsf{mix}[d-1]} \rangle$$

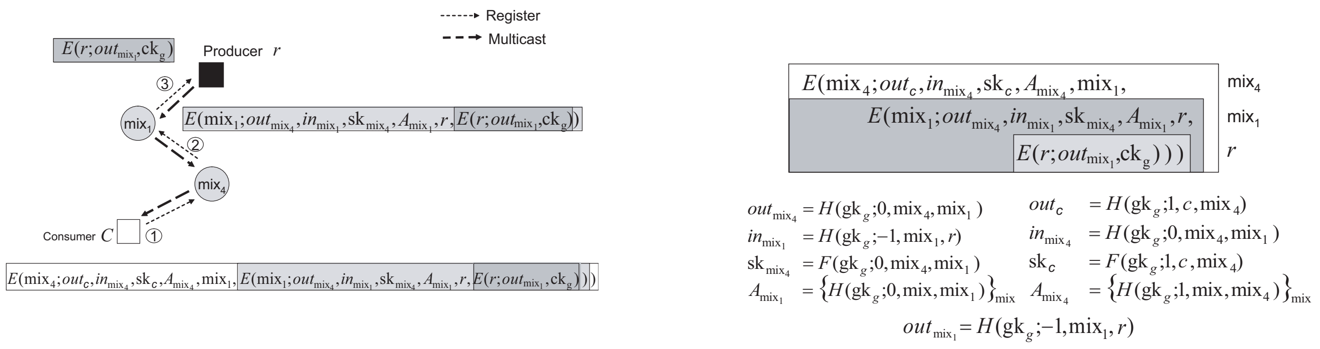
Each field informs the mix $\mathsf{mix}[d]$ how to process the registration message. More precisely, the fields are,

1. $out_{\mathsf{mix}[d+1]}$: the content label that $\mathsf{mix}[d]$ attaches when forwarding content to $\mathsf{mix}[d+1]$. $out_{\mathsf{mix}[d+1]} = H(\mathsf{gk}_g; d, \mathsf{mix}[d+1], \mathsf{mix}[d])$.

2. $in_{\mathsf{mix}[d]}$: the content label sent from $\mathsf{mix}[d-1]$. $in_{\mathsf{mix}[d]} = H(\mathsf{gk}_g; d-1, \mathsf{mix}[d], \mathsf{mix}[d-1])$. Note that $in_{\mathsf{mix}[d]} = out_{\mathsf{mix}[d]}$.

3. $\mathsf{sk}_{\mathsf{mix}[d+1]}$: the "step key" that $\mathsf{mix}[d]$ uses to encrypt content labeled with $in_{\mathsf{mix}[d]}$ before sending it to $\mathsf{mix}[d+1]$. This key is unknown to $\mathsf{mix}[d+1]$ who thus cannot read messages encrypted with this key (unless $d = h$, i.e., $\mathsf{mix}[d+1]$ is the consumer; see Section 5). $\mathsf{sk}_{\mathsf{mix}[d+1]} = F(\mathsf{gk}_g; d, \mathsf{mix}[d+1], \mathsf{mix}[d])$ where $F(\mathsf{gk}_g; \cdot)$ is a pseudorandom function keyed by gk_g with a sufficiently large range that the probability of collision is negligible.

[1]More properly, gk_g is used to pseudorandomly generate keys for use with multiple pseudorandom functions, though for simplicity of notation we will reuse gk_g in these pseudorandom functions directly.



(a) Traversal of a registration message through network.

(b) Registration message.

Figure 2. Example of Registration.

4. $A_{\mathsf{mix}[d]}$: the alias set. This field allows $\mathsf{mix}[d]$ to potentially aggregate the paths of two registration messages for the same producer. $A_{\mathsf{mix}[d]} = \{H(\mathsf{gk}_g; d, \mathsf{mix}, \mathsf{mix}[d])\}_{\mathsf{mix}\in\mathsf{Nbrs}}$ where Nbrs is a subset of $\mathsf{mix}[d]$'s neighbors containing $\mathsf{mix}[d+1]$. We analyze the size of $A_{\mathsf{mix}[d]}$ (i.e., the size of Nbrs) in Section 6.

5. $\mathsf{mix}[d-1]$: the next mix in the path to which $\mathsf{mix}[d]$ forwards the next encrypted layer, $z_{\mathsf{mix}[d-1]}$.

For $\mathsf{mix}[0]$, $z_{\mathsf{mix}[-1]}$ is simply the encrypted registration message for the producer, i.e., $E(r; out_{\mathsf{mix}[0]}, \mathsf{ck}_g)$. An example registration message is shown in Figure 2. In this example, a consumer c is interested in content generated by producer r. The consumer decides to route through two mixes, mix_4 and mix_1, to reach the producer. The registration message generated by c is shown in Figure 2(b).

4.2. Mix Processing

Each mix maintains a routing table which is updated with registration messages it processes. When a mix receives a registration message, the mix uses its routing table to determine if it has previously processed a registration message from the same (unknown) group. If so, the mix updates its routing table to reflect this added interest and drops the registration message. If not, the mix creates a new entry in its routing table and forwards the registration message to the next mix in the path.

Each entry in the routing table contains the fields "Lookup Label", "Aliases", and "Interested Neighbors." When a mix $\mathsf{mix}[d]$ receives a registration message $z_{\mathsf{mix}[d]}$, it decrypts $z_{\mathsf{mix}[d]}$ to obtain $\langle out_{\mathsf{mix}[d+1]}$, $in_{\mathsf{mix}[d]}$, $\mathsf{sk}_{\mathsf{mix}[d+1]}$, $A_{\mathsf{mix}[d]}$, $\mathsf{mix}[d-1]$, $z_{\mathsf{mix}[d-1]}\rangle$. For every entry e in its routing table, $\mathsf{mix}[d]$ computes $A_{\mathsf{mix}[d]} \cap A_e$ where A_e is the label set in the "Aliases" field for entry e. If $A_{\mathsf{mix}[d]} \cap A_e \neq \emptyset$, $\mathsf{mix}[d]$ adds $\langle \mathsf{mix}[d+1], out_{\mathsf{mix}[d+1]}, \mathsf{sk}_{\mathsf{mix}[d+1]}\rangle$ to the "Interested Neighbors" field in entry e; modifies A_e such that $A_e \leftarrow A_e \cup A_{\mathsf{mix}[d]}$; and drops $z_{\mathsf{mix}[d-1]}$. However, if every entry e results in $A_{\mathsf{mix}[d]} \cap A_e = \emptyset$, $\mathsf{mix}[d]$ creates a new entry with $\langle in_{\mathsf{mix}[d]}, A_{\mathsf{mix}[d]}, \langle \mathsf{mix}[d+1], out_{\mathsf{mix}[d+1]}, \mathsf{sk}_{\mathsf{mix}[d+1]}\rangle\rangle$ in the routing table and forwards $z_{\mathsf{mix}[d-1]}$ to $\mathsf{mix}[d-1]$. For the last mix $\mathsf{mix}[0]$ on the registration path, $\mathsf{mix}[-1]$ is replaced with r, the producer in which the consumer has interest.

An example of paths merging at a mix is shown in Figure 3. In Figure 3(a), mix_3 sends a registration message to mix_1. As mix_1 does not recognize any of the labels in mix_3's request, mix_1 creates an entry in its routing table and forwards the request to the next hop, the producer. In Figure 3(b), when mix_2 sends a request for the same topic to mix_1, mix_1 recognizes the overlapping label, and thus simply adds mix_2 to the entry and drops the registration request.

4.3. Producer Processing

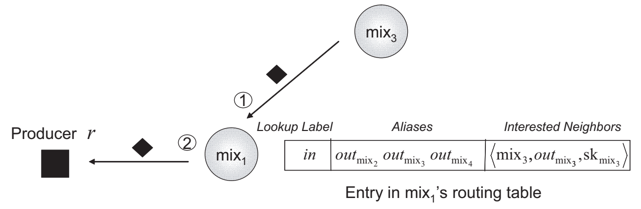
Each producer has a routing table similar to that of a mix; each entry in a producer's table consists of three fields: "Outgoing Label", "Content Key" and "Interested Mix." When a producer receives a registration message $\langle out_{\mathsf{mix}[0]}, \mathsf{ck}_g\rangle$ from $\mathsf{mix}[0]$, he creates a new entry and fills in the corresponding fields.

5. Multicasting Content

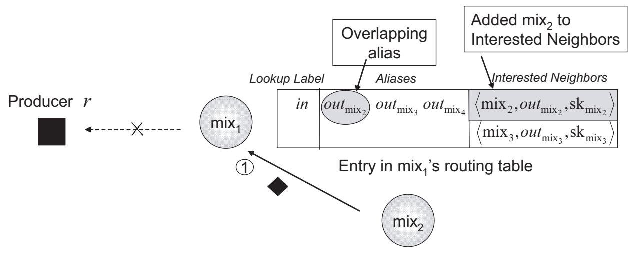
In the previous section, we described how a consumer registers his interest to a particular producer. In this section, we describe how content reaches its intended consumers.

5.1. Producer Steps

In order to multicast content M, producer r sends $\langle out_{\mathsf{mix}}, E(\mathsf{ck}_g; M)\rangle$ to each mix mix from which r directly received a registration message. Here, ck_g and out_{mix} are the content key and the label, respectively, and $E(\mathsf{ck}_g; \cdot)$ represents encryption with key ck_g.


mix3
①
Producer r
②
mix1
Lookup Label
Aliases
Interested Neighbors
in
out_{mix_2} out_{mix_3} out_{mix_4}
$\langle mix_3, out_{mix_3}, sk_{mix_3}\rangle$
Entry in mix1's routing table

(a) mix_3 sends a registration message to mix_1 who records mix_3 as an Interested Neighbor for this content.


Overlapping alias
Added mix2 to Interested Neighbors
Producer r
mix1
Lookup Label
Aliases
Interested Neighbors
in
out_{mix_2} out_{mix_3} out_{mix_4}
$\langle mix_2, out_{mix_2}, sk_{mix_2}\rangle$
$\langle mix_3, out_{mix_3}, sk_{mix_3}\rangle$
①
Entry in mix1's routing table
mix2

(b) mix_1 recognizes mix_2's register request is the same as mix_3's and records mix_2 as another Interested Neighbor for this content.

Figure 3. Example of paths merging at a mix.

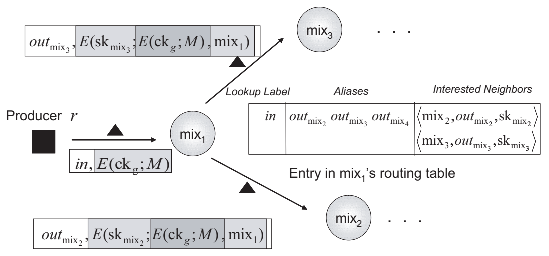

$out_{mix_3}, E(sk_{mix_3}; E(ck_g; M), mix_1)$
mix3
. . .
Lookup Label
Aliases
Interested Neighbors
Producer r
mix1
in
out_{mix_2} out_{mix_3} out_{mix_4}
$\langle mix_2, out_{mix_2}, sk_{mix_2}\rangle$
$\langle mix_3, out_{mix_3}, sk_{mix_3}\rangle$
$in, E(ck_g; M)$
Entry in mix1's routing table
mix2
. . .
$out_{mix_2}, E(sk_{mix_2}; E(ck_g; M), mix_1)$

Figure 4. Multicast.

5.2. Mix Processing

When a mix receives a multicast message, the mix uses its routing table to decide how to process the message. If there are multiple mixes who wish to receive the message, the mix copies the message and, based on the information stored in the routing table, tailors the message for each interested mix.

More specifically, for every content message $\langle \ell, b\rangle$ that a mix mix receives, mix processes the message as follows:

1. It inspects its routing table to find the entry e for which the lookup label in_e is equal to the message's label, ℓ.
2. For every tuple $\langle \mathsf{mix}', out_{\mathsf{mix}'}, \mathsf{sk}_{\mathsf{mix}'}\rangle$ found in the interested neighbor field of e, mix does the following:
   (a) Appends its own identity mix to payload b and encrypts the concatenation to create a new payload $b' = E(\mathsf{sk}_{\mathsf{mix}'}; b, \mathsf{mix})$.
   (b) Sends $\langle out_{\mathsf{mix}'}, b'\rangle$ to mix'.

For the last mix on the message path, mix' is replaced with the consumer's address, c. The use of link-specific encryption keys (sk is specific for each mix pair) to encrypt content is of particular importance. Because the keys used by mix_3 to encrypt messages destined for two mixes mix_1 and mix_2 are different, the ciphertexts corresponding to the messages that mix_1 and mix_2 receive are therefore different. This prevents mix_1 and mix_2 from deducing that they have received the same message from mix_3 by inspecting the payload of the message. We discuss the importance of hiding this information from the mixes in Section 6.

We note that the size of a multicast message grows as the message passes through the mix network. However, using techniques similar to those in Babel [9], a content message may pass through the network without growing in size.

5.3. Consumer Processing

When a consumer c receives a message $\langle \ell, b\rangle$ from the mix network, he repeatedly decrypts b to retrieve the actual content M. Since c knows the identity of the last mix, $\mathsf{mix}[h]$, on the message's path, c can generate the key, $\mathsf{sk}_c = F(\mathsf{gk}_g; h, c, \mathsf{mix}[h])$, used to encrypt b. Once b is decrypted, c learns the identity of $\mathsf{mix}[h-1]$. Using $\mathsf{mix}[h-1]$ and $\mathsf{mix}[h]$, c can generate the next key to decrypt the message and so on, thereby recovering the content M.

6. Analysis of M2

In this section, we analyze the effectiveness of M2 as an anonymous multicast communication system. The two metrics we use to evaluate M2 are the anonymity set size and the number of messages generated per content distribution. We remind the reader that the adversary is a global observer in control of the producer and some set of "honest but curious" mixes who wish to determine the groups which have interest in a producer's content. Additionally, we allow the adversary to have *a priori* knowledge of each group's size and the popularity (number of groups, number of consumers) of each producer's content. While in some cases it is not obvious how the adversary would obtain this information, permitting this knowledge to be public yields a conservative analysis.

6.1. Anonymity Set Size

A common method for evaluating mix-based systems is to calculate the sender effective anonymity set size of messages in the network (e.g., [6, 14]). A content message's producer anonymity set size is the entropy of the probability distribution of the random variable, R, the producer of the message. We remind the reader that the producer is controlled by the adversary, and so hiding the producer, per se, from the adversary is not our goal in M2. However, for unlinkability it is necessary that the adversary be unable to identify the producer to which a consumer subscribes. So, for a content message sent from a mix to a consumer (i.e., the "last hop" to the consumer), the only possibility of achieving unlinkability is to ensure that the content message has a large *producer* anonymity set size, even against an adversary who controls the producer who sent the content. It is similarly necessary to ensure a large *consumer group* anonymity set size for a content message sent from the producer to the first mix, though this analysis is the mirror of the producer anonymity set size. As such, here we illustrate the calculation of the producer anonymity set size only.

For a given content message m, let R be a random variable with a probability distribution function, P_{R}. $P_{\mathrm{R}}(r)$ is the *a posteriori* probability that m was sent by producer r, given the adversary's view of the system. The effective anonymity set size for m at any mix is calculated as:

$$S_m = -\sum_r P_{\mathrm{R}}(r) \cdot (\log(P_{\mathrm{R}}(r))).$$

To determine an effective anonymity set size, we must first calculate the producer probability distributions. We first explain how the producer probability distribution is computed for a non-corrupt mix. We then show how the producer probability distribution is affected by corrupt mixes in a traditional unicast mix system and in M2.

In a traditional, batching, unicast mix system, a mix has the same number of incoming and outgoing messages. Assuming that the adversary does not have control of the mix and the mix performs perfect mixing (i.e., each incoming message has equal likelihood of being any outgoing message), an adversary can calculate any producer's probability of producing any outgoing message as follows: denote $P_{\mathrm{R},1} \ldots P_{\mathrm{R},n}$ as the producer probability distributions for the n incoming messages. Let $P_{\mathrm{R},o}$ denote the producer probability distribution of an outgoing message. Then, the probability that an outgoing message is sent by producer r is:

$$P_{\mathrm{R},o}(r) = \frac{1}{n} \cdot \sum_{j=1}^{n} P_{\mathrm{R},j}(r) \tag{1}$$

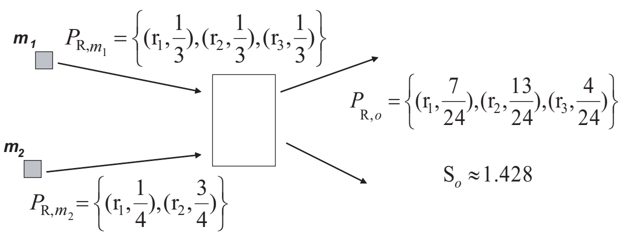
The entire producer probability distribution is computed by calculating $P_{\mathrm{R},o}(r)$ for each producer r. Given the producer probability distribution for an outgoing message, we can calculate the effective anonymity set size. As a simple example, consider a scenario where two messages m_1 and m_2 enter a mix, and that based on the adversary's view, each of m_1 and m_2 have only one potential producer, r_1 and r_2, respectively. Because the mix performs perfect mixing, the likelihood of r_1 or r_2 being the producer of either of the outgoing messages is $\frac{1}{2}$, and the producer probability distribution for each outgoing message is identical. An example in which m_1 and m_2 have multiple potential producers is shown in Figure 5(a).

The producer probability distribution calculation is much simpler for messages that enter a corrupt mix. Since the adversary controls the mix, each outgoing message can be correlated with an incoming message. Therefore, the producer probability distribution for each outgoing message is simply the producer probability distribution for its associated incoming message. As long as there is one non-corrupt mix in a message path, the producer of the message still remains anonymous.

In M2, however, corrupt mixes can reduce the anonymity set size of messages that are processed through the corrupt mixes. An M2 mix may output more messages than it receives due to multicast. If an adversary has *a priori* information of the number of consumers for a producer's content, knowledge that a message branches into multiple messages can help the adversary determine the likely producer of a message, thus reducing the effective anonymity set size. An example is shown in Figure 5(b).

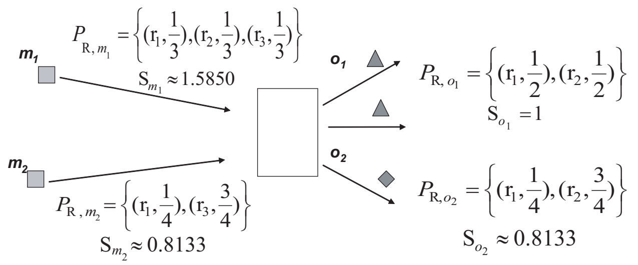
In Figure 5(b), there are three producers, r_1, r_2, and r_3. The adversary knows that r_3's content is not particularly popular (i.e., there are not multiple consumers for this producer's content), and therefore the probability that r_3 is the producer for a message m that branches into multiple messages is zero (i.e., $\Pr[\mathrm{R} = r_3 | m \text{ branches}] = 0$). Given two input messages m_1 and m_2, with producer probability distributions, P_{R,m_1} and P_{R,m_2}, respectively, an adversary who knows that m_1 branches can calculate the producer probability distributions of the corresponding outgoing messages. Since he knows that r_3 cannot be a producer for a message that has multiple consumers, the effective anonymity set size of the outgoing messages is lower than the effective anonymity set size of the corresponding incoming message.

We have shown that the amount of information gained by a corrupt mix is related to the number of times a message branches at the mix. We define the number of times a content message branches at a mix as its *fan-out*. Two factors affect the fan-out of a message at a particular mix: the number of registration messages that reach a mix and the probability that the mix can recognize that the registrations are from the same group. We first analyze the probability that a mix recognizes that registration messages are from the same group. We then analyze how different mix topologies may affect this ability.



m1
m2
$P_{R,m_1} = \{(r_1,\frac{1}{3}),(r_2,\frac{1}{3}),(r_3,\frac{1}{3})\}$
$P_{R,m_2} = \{(r_1,\frac{1}{4}),(r_2,\frac{3}{4})\}$
$P_{R,o} = \{(r_1,\frac{7}{24}),(r_2,\frac{13}{24}),(r_3,\frac{4}{24})\}$
$S_o \approx 1.428$

(a) Producer probability distribution with unicast.



m1
m2
o1
o2
$P_{R,m_1} = \{(r_1,\frac{1}{3}),(r_2,\frac{1}{3}),(r_3,\frac{1}{3})\}$
$S_{m_1} \approx 1.5850$
$P_{R,m_2} = \{(r_1,\frac{1}{4}),(r_3,\frac{3}{4})\}$
$S_{m_2} \approx 0.8133$
$P_{R,o_1} = \{(r_1,\frac{1}{2}),(r_2,\frac{1}{2})\}$
$S_{o_1} = 1$
$P_{R,o_2} = \{(r_1,\frac{1}{4}),(r_2,\frac{3}{4})\}$
$S_{o_2} \approx 0.8133$

(b) Producer probability distribution with multicast.

Figure 5. Example calculations of producer probability distributions.

6.1.1 Merging Paths

Recall that a mix's ability to coalesce paths is through the alias set in a registration message. The number of aliases in an alias set for a mix is thus bounded from above by the number of neighbors that the mix has. Let a be the alias set size, and $q > a$ be the number of neighbors a mix has. Assuming each group member forms an alias set seen at mix by choosing a mixes from among its q neighbors uniformly at random, the probability that mix can recognize two registration messages are for the same producer is:

$$\begin{aligned}\Pr[\text{paths merge}] &= 1 - \Pr[\text{alias sets disjoint}] \\ &= 1 - \frac{\binom{q-a}{a}}{\binom{q}{a}} \qquad (2)\end{aligned}$$

We can estimate the largest fan-out at a mix as a function of both the number of paths that cross at a mix and the probability that any two paths merge, using random graph theory. A random graph $\Gamma_{n,p}$ is an undirected graph on n vertices, where each possible edge is included independently with probability p. For a fixed mix, we form a random graph with the vertices denoting the paths that include mix at the same distance from the producer, and where the probability p of inserting an edge between two such vertices is simply the probability the two corresponding paths merge (Equation 2). Given this, each connected component in $\Gamma_{n,p}$ corresponds to one routing table entry in mix; i.e., for each component in $\Gamma_{n,p}$, an incoming message branches the number of times equal to the size of the component. Each connected component corresponds to the paths that merged into one path on which content is now received.

Given this, the maximum fan-out at mix corresponds to the size of the largest connected component in $\Gamma_{n,p}$. It is well-known (e.g., see [10, 1]) that the maximum component size is $O(\log n)$ in expectation if $pn < 1$, and otherwise is $O(n)$ in expectation (and all other components have size $O(\log n)$ in expectation). Figure 6 shows the expected maximum component size (fan-out) as a function of p for various numbers of paths (n). Each point is computed by averaging the results of ten simulations.

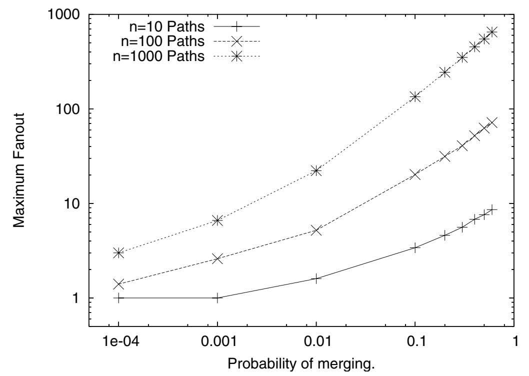


Figure 6. Average maximum fan-out as a function of the probability of paths merging.

As shown, the larger the number of registration messages from the same group that meet at a mix at the same distance from the producer, the larger the maximum component (fan-out) the mix sees. Since n can be influenced by the mix topology, we discuss particular topologies below.

6.1.2 Mix Topologies

In this section, we analyze the fan-out from two popular mix configurations: mix cascades [12] and fully-connected mix networks.

Mix Cascades In a mix cascade [12], all messages follow the same path through the mix network. In unicast systems, mix cascades can offer strong anonymity, and in particular are immune to certain attacks (e.g., intersection attacks) to which fully-connected mix networks are vulnerable [3].

However, if M2 is deployed on mix cascades, all registration messages from a group will arrive at the mix furthest from producers bearing the same alias set (of size one), i.e., the probability p with which these registrations will merge is $p = 1$. That is, all registration messages for a particular content merge at this mix, and so this mix becomes the multicast point for all messages (i.e., for each group, n equals

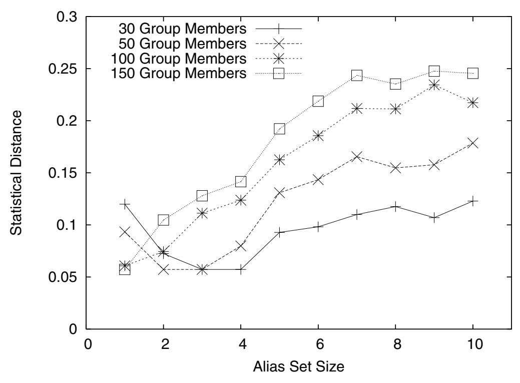


Figure 7. Statistical distance versus a baseline group with 10 members and alias set size of 10. $N = 30$ mixes, path length $= 10$.

the number of consumers for the topic). Since the fan-out seen at this mix corresponds directly to the size of the group, if this mix is corrupt and knows the *a priori* knowledge of each producer's content popularity, then it can narrow in on the content producer to which a group has registered.

Fully-Connected Mix Network In a fully-connected mix network, each registration message traverses a path through the mix network that the consumer chooses (we assume) uniformly at random (from among all paths of the specified length). As such, the number of registration messages n (and hence, the maximum fan-out, see Section 6.1.1) seen at any mix does not necessarily indicate the size of the group interested in the topic. However, the distribution of fan-outs seen by a mix for different groups is still dependent on the size of the groups. As such, witnessing a fan-out may still reveal the message's producer.

In order to reduce the amount of information gained by observing differing fan-outs at a mix, it is necessary to tune our system so that it is unlikely for a mix to see widely varying fan-outs. Specifically, let the random variable F_g be the fan-out that is witnessed at a mix for content for which members of an (unknown) group g registered. If there is a value f such that $\Pr[\mathrm{F}_g = f] \gg \Pr[\mathrm{F}_{g'} = f]$, then if an adversary-controlled mix witnesses a fan-out of f, the adversary can deduce that these registration messages were more likely generated by members in g than g'.

As such, our goal is to minimize the *statistical distance* between the distributions of F_g and $\mathrm{F}_{g'}$ for registration messages generated by g and g', where the statistical distance between the probability distributions of two random variables X and X' over the same space $\mathbb{X}$ is:

$$d(\mathrm{X}, \mathrm{X}') = \sum_{x \in \mathbb{X}} |\Pr[\mathrm{X} = x] - \Pr[\mathrm{X}' = x]| \,.$$

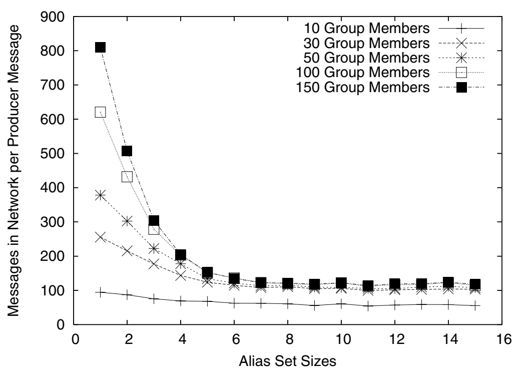
The statistical distance between two probability distributions measures the "closeness" of the two distributions. By minimizing this distance for the distributions of F_g and $\mathrm{F}_{g'}$ (henceforth, the *fan-out statistical distance*), we make it less likely that fan-outs will disclose information about the group to which they correspond. Conversely, a large fan-out statistical distance between groups will make it easier for corrupt mixes to distinguish messages destined for groups with known, different sizes. An example is shown in Figure 7, which shows the fan-out statistical distance between a baseline group with 10 members and other larger groups. The alias set size for the baseline group is set to 10. As shown, the largest statistical distance occurs when all groups set their alias set sizes equal to 10. Moreover, for the same alias set size, the fan-out statistical distance from the 10-member group increases as the number of group members increases. This follows from Section 6.1.1 as larger fan-outs occur when there are more members in a group.

One way to reduce the fan-out statistical distance between groups is to reduce the alias set size a of the larger groups. From Equation 2, the smaller a is, the less likely paths that meet at a mix will merge. By reducing a, we effectively reduce the probability that large fan-out occurs at any particular mix. From Figure 7, we verify that using smaller alias set sizes for larger groups indeed reduces the fan-out statistical distance. This demonstrates that by tuning the alias set sizes for all groups in the network, M2 can provide multicast capabilities while preventing an adversary from using fan-out as a method to gain significant knowledge.

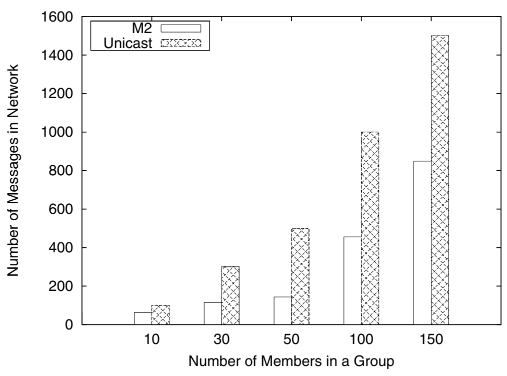
One caveat exists to our solution: a corrupt mix that processes a registration message with a reduced alias set can recognize that the message generates a path for a larger group. As such, after the consumer populates the alias set for a mix with a elements, it can "fill out" the alias set to a predetermined system-wide size v by adding $v - a$ elements chosen randomly from the range of the pseudorandom function $H(\cdot;\cdot)$. In this way, the alias set sizes of all registration messages will appear to be of equal size v.

6.2. Message Savings

A consequence of reducing alias set sizes for large groups as described in Section 6.1 is an increase in the number of messages sent in the network to convey content to consumers. Indeed, alias set size is the primary tool we have for reducing message costs, as shown in Figure 8(a). This figure shows that the number of messages in the network for a content distribution decreases dramatically as the alias set size grows. For example, a group of 150 consumers could have over 80% message savings over unicast distribution when the alias set size is set to 10. It is also interesting to note that when the alias set size is large, the number of messages generated in the mix network for a group with 150 group members is similar to those for a group with 30 group



(a) Various alias set sizes.



(b) Alias set sizes chosen to minimize statistical distance against group size 10 with alias set size equal to 10.

Figure 8. Number of messages generated with N=30, path length = 10.

members. This is due to the significant savings from multicast that arises from the large fan-outs for the large groups. However, this difference in fan-out is exactly the reason for reducing alias set sizes for large groups when small groups are also present, as discussed in Section 6.1.

Reducing alias set sizes for larger groups raises the possibility that message savings will be reduced to an extent that renders multicasting of limited utility. However, our experiments demonstrate continued, significant message savings even when alias set sizes are adjusted to minimize the fan-out statistical distances between groups. A representative graph is shown in Figure 8(b), which depicts the number of messages generated using alias set sizes that minimized the fan-out statistical distance from a group with 10 members in Figure 7 for a system with $N = 30$ mixes. As shown, despite the reduced alias set sizes, the number of messages generated per producer message is significantly smaller than sending content messages via unicast.

To demonstrate the trade-off between message savings and anonymity, we take representative group sizes from Figures 7 and 8(b) and, for various alias set sizes per group, plot both the number of messages saved versus unicast per producer message and the fan-out statistical distance from a group with 10 members. The graph is shown in Figure 9. As expected, as alias set size increases, the number of messages saved increases but at the cost of increased fan-out statistical distance. This graph shows that given a requirement of either minimum number of messages to be saved or maximum fan-out statistical distance allowed, we can estimate the impact that one requirement has on the other. As an example, in Figure 9, if a group of 150 members is willing to tolerate a fan-out statistical distance of 20% from the baseline of a 10 member group, then 800 messages could be saved over unicast distribution of content.

Another parameter that affects the message complexity is the network size N. Recall from Section 6.1 that the probability of merging is dependent on the number q of neighbors a mix has. For the fully connected network, every mix is a neighbor of every other mix, i.e., $N - 1 = q$. Figure 10 shows the effect the network size has on the number of messages generated for a group of size 30. As shown, network size has a significant impact on the number of messages generated in network: as N grows, the opportunities to merge paths decreases, and so the number of messages saved also decreases. However, Figure 10 shows that this trend can be somewhat offset by increasing alias set size. As the amount of anonymity lost depends on the sizes of existing groups in the system, careful tuning is necessary to balance the trade-off between message savings (even in larger networks) and anonymity.

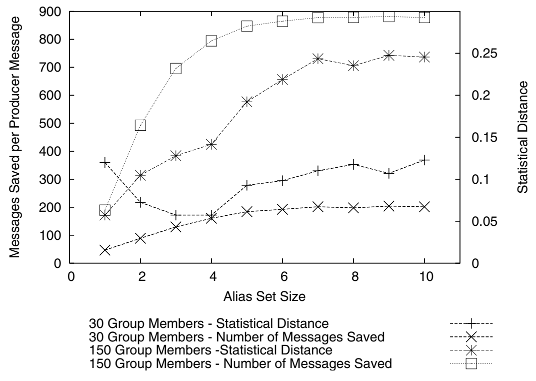


Figure 9. Messages saved vs. unicast, and fan-out statistical distance from a group size of 10 with alias set size equal to 10, versus differing alias set sizes for other group sizes.

Our experiments show that multicasting offers significant message savings even when tuned to maximize the anonymity protection it can afford. That said, we anticipate that for systems supporting groups with widely disparate sizes, it may be more appropriate to partition groups into

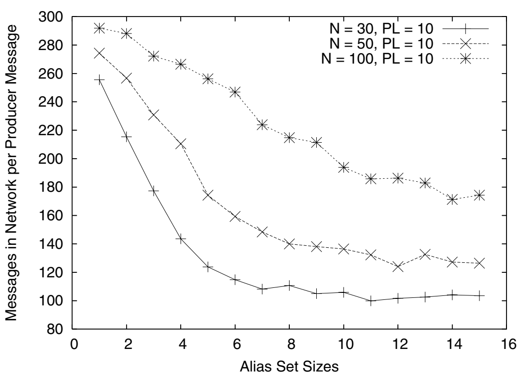


Figure 10. Messages generated with group members = 30 and path length = 10.

classes (e.g., "small", "medium" and "large" sizes) and adjust alias set sizes so as to minimize the fan-out statistical distances between groups within each class. This will result in near optimal message savings that multicasting can provide, at the cost of revealing to corrupt mixes the class of the consumer group (but not the particular group to which it pertains).

7. Conclusion

In this paper, we presented a novel anonymous multicast protocol, M2, which allows producers to multicast content to consumers while providing unlinkability against a global adversary in control of both the producer and some set of mixes. Mutually trusting consumers join consumer groups to register interest for content through the mix network, thereby allowing mixes in the network to opportunistically "merge" the paths of consumers in the same group. Our analysis showed that in certain mix configurations, multicast can degrade unlinkability. In particular, we showed that a traditional mix network architecture that provides strong unicast unlinkability, mix cascades, does not necessarily provide strong unlinkability when multicast is employed. We also demonstrated that in a fully-connected mix network, it is feasible to provide multicast services while still retaining unlinkability. Specifically, we showed that when group sizes are large and similar, we can achieve 80% message savings without loss in unlinkability. In scenarios where groups have vastly different sizes, we showed that message savings must then be traded-off to reduce the fan-out statistical distance between the groups. In such cases, the number of message generated was still reduced by a factor of two over unicast distribution. We conclude that with careful tuning of M2 parameters, M2 can provide the benefits of multicast while mitigating unlinkability degradation due to multicast.

References

[1] R. Z. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[2] T. Ballardie. Scalable multicast key distribution. IETF RFC 1949, 1994.

[3] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free mix routes and how to overcome them. In *Privacy Enhancing Technologies Workshop*, 2000.

[4] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Eurocrypt*, 1994.

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

[6] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *Privacy Enhancing Technologies Workshop*, 2002.

[7] E. Gabber, P. B. Gibbons, Y. Matias, and A. Mayer. How to make personalized web browsing simple, secure, and anonymous. In *Financial Cryptography*, 1997.

[8] C. Grosch. Framework for anonymity in ip-multicast environments. In *Globecom*, 2000.

[9] C. Gulcu and G. Tsudik. Mixing email with Babel. In *Network and Distributed Systems Security*, 1996.

[10] S. Janson, T. Luczak, and A. Ruci. *Random Graphs*. Wiley and Sons, 2000.

[11] D. Mazieres and M. F. Kaashoek. The design, implementation and operation of an email pseudonym server. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 27–36, 1998.

[12] A. Pfitzmann, B. Pfitzmann, and M. Waidner. Isdn-mixes - untraceable communication with very small bandwith overhead. In *GI/ITG Conference on Communication in Distributed Systems*, 1991.

[13] A. Pfitzmann and M. Waidner. Networks without user observability. *Computers & Security*, 2(6):158–166, 1987.

[14] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies Workshop*, 2002.

[15] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In *Computer and Communication Security*, 2000.

[16] N. Weiler. Secure anonymous group infrastructure for common and future internet applications. In *Computer Security Applications Conference*, 2001.

[17] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communication using key graphs. In *ACM SIGCOMM*, 1998.