Mixing Email with BABEL

Ceki Gülcü Gene Tsudik
IBM Research Division, Zürich Research Laboratory
Säumerstrasse 4, CH-8803 Rüschlikon
Switzerland
email: gts@zurich.ibm.com
tel +41.1.724-8308 fax +41.1.710-3608

Abstract

Increasingly large numbers of people communicate today via electronic means such as email or news forums. One of the basic properties of the current electronic communication means is the identification of the end-points. However, at times it is desirable or even critical to hide the identity and/or whereabouts of the end-points (e.g., human users) involved.

This paper discusses the goals and desired properties of anonymous email in general and introduces the design and salient features of Babel anonymous remailer. Babel allows email users to *converse* electronically while remaining anonymous with respect to each other and to other – even hostile – parties. A range of attacks and corresponding countermeasures is considered. An attempt is made to formalize and quantify certain dimensions of anonymity and untraceable communication.

Keywords: security, email, mix, anonymity, untraceability, traffic analysis, remailer

1 Introduction

Explosive growth and proliferation of the global Internet in the past decade allowed millions of people to communicate via electronic mail. In many respects, email is rapidly replacing traditional paper mail. Email is not only fast and convenient but also – at least for the time being – free of charge for a large segment of users.

There are, however, some aspects of email that can be improved upon. First, most of today's Internet email is not very secure. Sender authentication, non-repudiation, data integrity and privacy are some of the basic ingredients of secure email. While basic email security is addressed to some extent by recent offerings such as PGP [35] and PEM [16], their acceptance is far from universal. Another important feature missing in current email is support for anonymity and untraceability of users. In the Internet milieu, it is quite unrealistic to expect any security features of the underlying network; eavesdroppers can easily record email messages and gather addressing information. Traditional paper mail, in contrast, allows one to send an envelope with a printed destination address and no return address. This, coupled with other common-sense precautions, can make the sender untraceable and anonymous; police and sleuth fiction to the contrary notwithstanding.

In this paper we discuss the goals and desired properties of anonymous email and then describe the design and features of the BABEL– an anonymous remailer developed at IBM Zurich Research Laboratory. In brief, our approach is based on a special entity called a "mix". The concept of a mix was first introduced by Chaum [2] in the early eighties. A mix can be viewed as a logical component (e.g., application layer software) that forwards email messages and – in the process – obfuscates the relationship between incoming and outgoing message traffic.

The paper is organized as follows. In the next section we begin by motivating the need for anonymity, briefly reviewing previous work and describing the goals of anonymous/untraceable email. Then, in Section 3 we introduce the concept of a mix and consider the threats it faces. Sections 4, 5 and 6 are devoted to the technical discussion of BABEL anonymous remailer. Section 7 presents an attempt to quantify some measures of anonymity. Finally, Section 8 describes the salient implementation issues.

2 Motivation

It is no surprise that untraceable communication is a highly-charged and, at times, even controversial, topic [1, 18, 22, 23]. Anonymous email is an anathema to some people. This reputation is due largely to the possible abuses of anonymity for the purposes of spreading libelous accusations, hate-filled propaganda, pornography and other unpleasant content.

At the same time, anonymous mail has its legitimate and benign uses. We divide these into four main categories:[1]

1. Discussion of sensitive and personal issues
2. Information searches
3. Freedom of speech in intolerant environments
4. Polling/Surveying

Many people in need of counseling or therapy, such as victims of sexual, alcohol or drug abuse, can receive support and counseling electronically while remaining anonymous. For example, a victim of abuse would

[1]The list is not meant to be exhaustive.

probably be reluctant to participate in on-line therapy sessions if there was a chance that someone they knew was "listening". Thus it is often critical for the identity of the user to remain secret. This need is also widely recognized by the medical profession.

We often seek information anonymously in the course of our everyday life. For example, an employee of one company may inquire about a job opening at another (perhaps competing) company; the need for anonymity is obvious.[2] Furthermore, people often seek information from sources that, should the identity of the seeker become known, would act in a manner not agreeable to the seeker. For example, a consumer might like to browse a number of electronic shops and compare prices before making a purchase. If the consumer's identity were revealed, the visited shops could place his name/address on their mailing lists and start bombarding his mailbox with unwanted "junk" email. There are other everyday cases where anonymity is an integral part of a transaction.

On a more somber note, there are still, alas, a number of totalitarian regimes in the world; places where nonviolent (e.g., verbal) opposition or dissent can have serious consequences including imprisonment, torture and death. Furthermore, even in the free world, there are intolerant and fanatical groups that violently and virulently harass critics for mere opinions. Examples abound...

In the same vein, there are also many well-known situations in which an individual may feel compelled to report corruption, criminal behavior or other misdeeds. In such cases, being anonymous means being safe from varying degrees of retribution.

Another useful, albeit rather non-controversial, application of anonymous email is in the area of polling and surveying. There are a number of organizations specializing in opinion surveys on a wide variety of topics. Participants' anonymity is one of the basic features of this activity.

Admittedly, the fundamental motivation for hiding one's identification is the fear of retribution (either rightful or wrongful.) It is not the goal of this paper to partake in the currently on-going debate on privacy and anonymity on the Internet. We only note that anonymity is an optional (and mostly legal) part of regular, paper mail. Obviously, it can be misused, yet there are no great debates on banning anonymous usage of paper mail. Drawing a boundary between use and abuse of technology is a complicated philosophical matter; it is not treated in this paper.

2.1 Previous work

The first and the most authoritative paper to-date dealing with anonymous communication was published by D. Chaum in 1981 [2]. The BABEL remailer described in this report owes much to his ideas. Chaum also invented the DC-network [4] which provides unconditional untraceability commensurate with high bandwidth overhead. Pfitzmann and Waidner have also done a considerable amount of work on anonymity and untraceable communication in LAN and ISDN environments. [24, 26, 25, 27].

The oldest and (currently) most widely-used anonymous remailer is located in Finland. It is called *Penet* and is operated by J. Helsingius. Penet performs the following functions:

It strips off all header information of the incoming mail before forwarding it to its final destination. Then, if not already assigned, an alias for the sender is created. In the outgoing message, the address of the sender is replaced by an alias. The alias allows the recipient(s) of the message to reply to the real sender without knowing his identity.

The demand on the Penet remailer is quite high: over 7,000 messages are sent daily. The alias database contains 200,000 entries [11]. Recently, Penet has become the subject of some controversy[3].

The second brand of remailers are promoted by a group called *cypherpunks* [12]. There are about 20 publicly available *cypherpunk* remailers. These remailers offer some of the basic functionality described in this paper. Although they share the same code base, each differs in minor ways; some allow posting to newsgroups while others do not, some do not accept PGP encrypted messages; some even use different formats. Their lack of a unified *modus operandi* complicates their use and hinders their acceptance. The *Mixmaster* [7] remailer written by L. Cottrell is a significant step forward as it constitutes the first true mix.

2.2 Overview of desired properties

We begin the technical discussion by enumerating the desired properties of anonymous mail.

1. Anyone able to send email should be able to do so anonymously.

2. It should be impossible (or, at least, computationally hard) to determine the originator of anonymous mail.

3. The receiver(s) of anonymous mail can reply to the sender, who remains anonymous. Moreover, receiver(s) may reply with multiple messages. (It is important to note that someone replying to an anonymous message, by definition, sacrifices some anonymity because the original sender "knows" the intended receiver(s) and can correlate a reply with an earlier message.[4])

4. Individual remailers intervening in anonymizing messages should be trusted as little as possible.

[2] This example is of *proactive* job search; it is different from the usual *reactive* search whereby the job descriptions are broadcasted to the "masses", e.g., by posting in appropriate newsgroups.

[3] On February 8, 1995, based on a burglary report filed with the Los Angeles police, transmitted by Interpol, Finnish police presented Helsingius a warrant for search and seizure. Bound to do so by law, he complied, thereby revealing the electronic address of a *single* user.

[4] However, some degree of anonymity can be preserved. For example, a reply to an anonymous newsgroup post only reveals the newgroup to the original poster; the identity of the replying party remains secret.

The anonymity of the end-points should be preserved even if a number of intervening entities collude or are subverted.

5. The remailer infrastructure should be resistant to both passive and active attacks. (This property is elaborated on below.)

6. The sender of anonymous email can (anonymously) obtain confirmation that it has been properly processed by the remailer system.

7. Anonymous email should not overload the global email infrastructure. (For example, if anonymity requires generation of email *noise* its volume should be kept low.)

2.3 Notation

The following notation is used throughout the remainder of the paper:

M	message; sequence of ASCII bits
$E_x(M)$	encryption of M with X's public key
$D_x(M)$	decryption of M with X's private key
$K\{M\}$	conventional encryption of M with key K
(M_1, M_2)	concatenation of M_1 and M_2
$\mathcal{A}_x$	X's email address.
$\lceil M \rceil^{\Omega}$	padding string M to length Ω (by appending random bits)
$\lfloor M \rfloor^{\Omega}$	trimming string M to length Ω (by removing trailing bits)

3 MIX - fundamental building block

As already mentioned, an anonymous remailer, or a mix, is an entity that, in addition to forwarding incoming messages, strives to hide the relationship between incoming and outgoing message traffic. (See Figure 1.)

In our model we assume the existence of a powerful adversary – Eve – capable of recording, removing or altering packets entering or leaving a mix. Eve is also able to generate spurious messages.

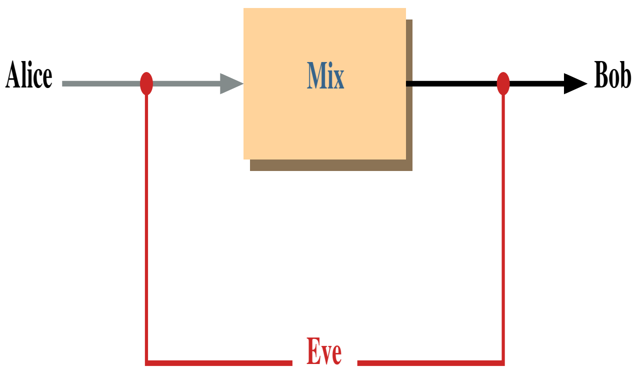

Figure 1: Basic Model

A mix functions according to the following principle [2]. Suppose Alice wishes to send message M to Bob anonymously. She submits a specially composed message I to the mix. I includes M and Bob's network address. It is intelligible only to the mix. A transformed version of I, called O, is forwarded by the mix to Bob. Ideally the relation between the incoming message, I, and the outgoing message, O, is obfuscated. Thus, Eve is unable to connect Alice to Bob. This kind of anonymity is called "unlinkability of sender and recipient"[27].

There are two ways for Eve to correlate incoming and outgoing messages: i) by contents, i.e., message data or message size, or, ii) by *causality*, i.e., by associating time of message arrival with that of its departure.

In general, content correlation can be addressed by using standard cryptographic techniques along with padding. Causal correlation can be easily countered if the incoming traffic volume is sufficiently high. In the next section we focus on making content and causal correlation difficult.

3.1 Passive attacks

This section addresses so-called passive attacks, i.e. those that can be carried out by merely observing message traffic.

3.1.1 Content correlation

Two elements can help in content correlation: actual content and length. For prevention it suffices that all messages to/from a mix be encrypted and be of uniform length. We denote this length by Ω.

The user encrypts his message M and the destination address $\mathcal{A}_{\mathrm{Bob}}$ with the mix's public key. Thus,

$I = E_{\mathrm{mix}}(\mathcal{A}_{\mathrm{Bob}}, M)$ where $\mathcal{A}_{\mathrm{Bob}}$ is Bob's network address.

Upon receipt and successful decryption of I, the string $(\mathcal{A}_{\mathrm{Bob}}, M)$ is be revealed. The output message O, consisting of M (in cleartext) and other data added by the underlying communications network, is forwarded to Bob at $\mathcal{A}_{\mathrm{Bob}}$. Eve may attempt to correlate O and I by comparing $E_{\mathrm{mix}}(\mathcal{A}_{\mathrm{Bob}}, M)$ and I. To outwit Eve, random one-time "salt" must factored into the encryption to ensure that successive encryptions of the same message yield different results.

In hybrid systems based on both public and conventional key encryption the random string might be unnecessary. Such systems typically use a random session key to encrypt user data with a symmetric key algorithm and a public key encryption algorithm to encrypt the random session key. Each encryption with a public key uses a different session key, which is revealed only to the owner of the private key (the mix in this particular case). Thus, Eve is unable to correlate I and O even though she is able to re-encrypt $(\mathcal{A}_{\mathrm{Bob}}, M)$. The re-encryption results in I', which bears no resemblance to I; refer to [26] for cryptographic attacks on straight-RSA implementation of mixes.

In order to avoid size correlation, message sizes must be constant throughout the entire mix network. Message size uniformity can be achieved by padding to a constant length (Ω) with random data. Although seemingly innocuous, padding is an important issue and greatly influences the implementation of a mix. A detailed discussion of this issue is postponed until Section 6.

Note that the security of the system is based on the integrity of a mix. In a single-mix architecture, if the mix is somehow forced to reveal its private key, identities of users can be compromised. Multiple mixes can be used to increase the security of the whole system. This is discussed in the following sections.

3.1.2 Time correlation

Obviously, there is a strong causal relationship between the incoming and outgoing messages. This relationship can be exploited by Eve. One simple solution is to output messages in batches, as outlined in [2]. In this scheme, at least N input messages are accumulated before being forwarded in random order. N is called the *minimum batch size*. We refer to this scheme as normal or regular batching.

Under low load conditions, incoming messages may be so scarce that a batch of size N cannot be formed within a reasonable time. Sending out random-looking *decoy* messages to random destinations solves (or at least alleviates) the problem. Decoys are indistinguishable from normal messages except that they are immediately discarded by their recipients after decryption.

In an enhanced scheme, called *interval batching*, we divide time into equal periods of length T. Let n be the number of incoming messages in a given period. The following procedure is performed at the end of each period:

normal batching	if $n \geq N$
$N - n$ decoys followed by batching	if $0 < n < N$

This approach guarantees that a message will be delayed at most T units of time by a mix. Note that batching messages introduces a risk because anonymity then depends on the behavior of other users. This external dependence can pave the way for other attacks (see Section 3.2.1.)

Another popular approach to solving the time correlation problem involves introducing a random delay for each message. This randomness makes the system nondeterministic but not necessarily safer. We avoid this venue.

3.2 Active attacks

In this section we discuss active attacks, i.e. those involving direct modifications to message flow, by altering, inserting, delaying and even deleting, messages.

3.2.1 Isolate & Identify

If regular batching is used, Eve may submit a number of messages to a mix, forming an almost complete batch, with only one message missing. Upon arrival of a genuine message, the entire batch is forwarded and Eve can simply pick out the message she did not generate [27]. Note that, although the genuine message may be encrypted, Eve is able to correlate the genuine message with its outgoing counterpart. The mix is thus considered defeated.

In the interval-based batching approach, flooding a mix is useless if genuine traffic is heavy. However, when few legitimate messages arrive in a given interval, flooding causes the mix to believe that decoys are unnecessary. Eve might even remove or rearrange messages so that one real message trickles into the mix per period. Then, by injecting false messages Eve is able to link the single authentic message with its outgoing counterpart.

This attack is difficult to thwart completely. One simple but only partial countermeasure is to require a certain number of decoys even when a batch is full. A more effective approach is the introduction of intermix detours; it is discussed in Section 5.6.1.

3.2.2 Message Replay

Eve can try to defeat a mix by recording a genuine message and reinserting it later into the message stream. As an incoming message I results in the same output O when replayed, associating the two is trivial. Because of its simplicity, message replay is an extremely serious threat. It is possible to prevent replay by keeping track of incoming messages and discarding replays [2]. Replay detection is a well-studied topic [8, 9]. Basic techniques consist of using sequence numbers, random numbers (nonces) or data and time stamps.

Techniques involving sequence numbers or nonces imply at least some synchronization. However, there is an inherent contradiction between the terms synchronization and anonymity. Moreover, traditional methods are concerned with authentication, which is not required in our case. Under these circumstances, we have decided to use a variant of a time-stamp scheme.

In brief, each message is uniquely identified and time-stamped. Clearly, the identifier should reveal no information about the message. Assuming the use of hybrid message encryption cryptosystem (e.g., as in PEM or PGP) we use the public key encrypted form of the session key as the message identifier. Since a message does not decrypt correctly even if a single bit of the encrypted session key is altered, it is an *invariant* of replays. The session key is unique with a high degree of probability because, it is usually generated at random from a very large key space[5]. This method is also very cost-effective since a mix does not have to perform any expensive operations to calculate unique identifiers for the incoming messages; it simply copies the encrypted session key.

It is certainly undesirable to keep track of messages indefinitely as it would result in excessive space usage. A simple solution is to time-stamp messages and flush message entries after some fixed system-wide time interval. This point is further discussed in Section 8.

Replying to messages is somewhat different because "replays" along a reply path are perfectly legitimate (see Section 5.6).

[5] PGP uses 128 bit IDEA-keys. Moreover, before RSA-encrypting this IDEA-key, it randomly pads it to the modulus of the public key.

3.3 Cascading or chaining mixes

We now assume that there is a pool of mixes at the users' disposal. As mentioned earlier, if only a single mix is used, that mix is trusted to withhold critical information. Instead of trusting a single mix, Alice may decide to use a series of mixes to forward her message to Bob [2], see Figure 2. The system thus becomes more secure.

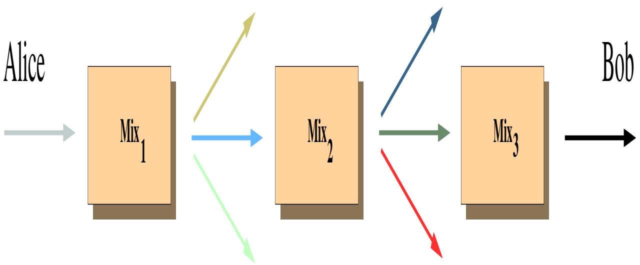

Figure 2: Chaining mixes

Eve's task becomes significantly more difficult. In fact, in order to link the message sent by Alice to the message received by Bob, Eve has to subvert/defeat the mixes on the path.

If Eve is a global observer of the mix network she can simply concentrate on messages entering and leaving the system without paying attention to inter-mix traffic. If the traffic load is low, the security degenerates to the worst-case scenario outlined in Section 3.1.2. However, in practice, a large number (> 100) of independent mixes distributed around the world would make it very difficult for Eve to be a global observer.

4 Forward Path

In this section we describe the process of generating anonymous messages and their subsequent handling at intervening mixes. Most of the material (with few exceptions) presented in this section is due to Chaum [2, 3, 4].

For the sake of clarity, we assume (for the time being) that cryptographic operations (encryption/decryption) have no impact on message size. We will return to this issue in Section 6.

4.1 Composition by sender

Suppose Alice wishes to send an anonymous message to Bob through f mixes; $F_1, F_2, \ldots, F_f$. This set of mixes is referred to as the *forward path*[6]. She composes her message according to the following procedure:

(1) The cleartext message is padded to exactly Ω bytes. The maximum allowed cleartext message size is, α, where $\alpha < \Omega$. This restriction ensures that each message is padded with at least $\beta = \Omega - \alpha$ random bytes. The reason for reserving β bytes for padding will become clear in Section 6. The parameters Ω and α are *system-wide* constants.

(2) The padded message $\lceil M \rceil^{\Omega}$ is then encrypted once for every mix on the forward path, starting with the last, F_f, is encrypted in the following manner:
$x_1 = E_{F_f}(\mathcal{A}_{\text{Bob}}, \lceil M \rceil^{\Omega})$

$x_i = E_{F_{f-i+1}}(\mathcal{A}_{F_{f-i+2}}, x_{i-1})$, for $1 < i \leq f$

where E_{F_i} represents public key encryption with mix F_i's key. The final outcome is:
$x_f =$
$E_{F_1}(\mathcal{A}_{F_2}, E_{F_2}(\ldots E_{F_{f-1}}(\mathcal{A}_{F_f}, E_{F_f}(\mathcal{A}_{\text{Bob}}, \lceil M \rceil^{\Omega}))\ldots))$

The result is analogous to an *onion* where each encryption is likened to a layer of skin. To access inner layers, outer layers must be stripped off first (see Figure 3.) The effect of encryption on message size is shown in the figure. In particular, the dimensions of the boxes show how message size increases with each encryption and concatenation step.

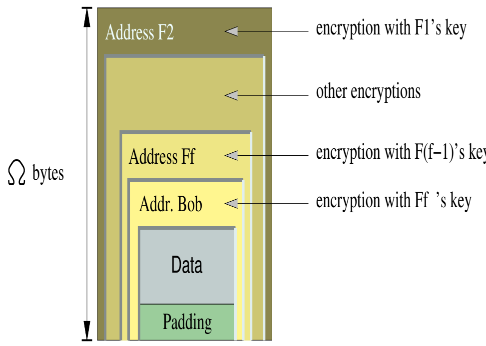

Figure 3: Forward message prepared by Alice

(3) Once the *onion* is assembled it is sent to the first mix on the forward path, F_1.[7]

Note that the encryption steps are all performed at the sender, the only trusted entity. No encryption takes place at the successive mixes. This ensures that the information revealed to each mix is kept to a minimum.

4.2 Processing by mixes

The first remailer, upon reception of Alice's message, decrypts it with its secret key to discover the address of the next hop, $\mathcal{A}_{F_2}$. This is analogous to removing the first of skin of the onion.

Similarly, each mix on the forward path removes a layer of encryption until the last hop is reached. The last remailer strips off the remaining layer and

[6] We use the letter F to denote mixes on the forward path.

[7] The binary data produced by encryption might be unsuitable for transmission as email. In that case, an appropriate format conversion must take place.

discovers $\mathcal{A}_{\text{Bob}}$. The message is then delivered with all padding removed.

The actual message received by Bob shows that it has been delivered by mix F_f. Bob does not know the identities of the other mixes nor that of the originator, Alice[8].

To accommodate the largest possible number of users, the system of remailers assumes that the recipient of anonymous messages has only basic email capability with no specific software to handle anonymous messages. Only regular mail is delivered to the final destination. In other words the last mix sees the message in cleartext.

If the destination has encryption capability (e.g., PGP) Alice can encrypt the message using the recipient's public key. Thus, the contents of the message are hidden from the last hop, and a higher degree of security is achieved. Obviously, if the destination is a public newsgroup, using secret keys make little sense.

4.3 What does a mix know?

One important security measure of the entire mix network is the amount of knowledge gained by a mix in the course of processing a message. By examining email-specific fields (e.g., SMTP headers) an intermediate mix on the forward path can discover the identity of the previous mix hop. Without some questionable hacking of email software it appears impossible to prevent a mix from gaining this knowledge.

Another piece of information visible to a mix is the identity of the next hop. It is possible – albeit in theory – to prevent an intermediate mix from knowing the next hop.[9] We briefly sketch one simple method:

Alice composes the anonymous message much as before but omits mix addresses – $\mathcal{A}_{F_i}$ – from the layers. Each intervening mix, instead of sending the message to the next hop, posts it to a newsgroup periodically scanned by all mixes. (An alternative is to broadcast the message to all mixes.) All mixes try to decrypt the message but only one succeeds. The same procedure is repeated until the last mix is reached; the last mix forwards the message directly to Bob.

Although it *halves* the knowledge gained by intermediate mixes this solution is fraught with difficulties: the performance overhead alone would be staggering. A more practical, but commensurately scaled down, variation is to give the sender an option to include multiple mix addresses in each layer. This way, an intermediate mix forwards an outbound message to several next-hop mixes and remains uncertain with respect to the identity of the actual next hop.

5 Return Path

Thus far we discussed how to send messages anonymously without enabling replies. Although uni-directional communication is most amenable to anonymity, it is sometimes desirable for an anonymous mail recipient to reply to the (still anonymous) sender. This can be achieved by giving the sender an option of including a Return Path Information (RPI) in the anonymous message.

5.1 Creating the RPI

The RPI is composed by Alice according to the following procedure.

(1) Alice chooses mixes $R_1, R_2, \ldots, R_r$ for the return path and mixes $F_1, F_2, \ldots, F_f$ for the forward path. (See Figure 4.)

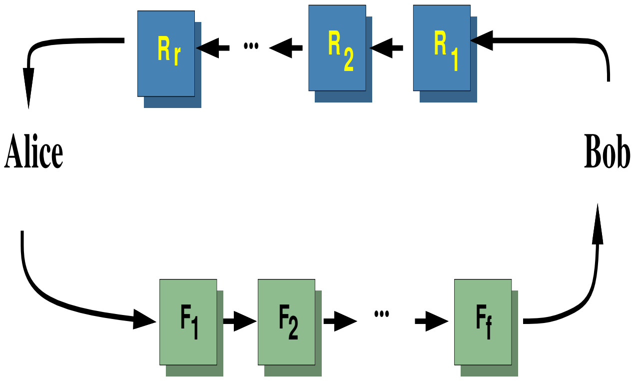

Figure 4: Return Path

The mixes on the forward path and return path are completely independent. The two sets may be identical, overlapping or completely disjoint.

(2) Alice randomly chooses a *key seed* – KS – and, using it, computes r keys, $K_1, K_2, \ldots, K_r$. There are many ways to do so, e.g.: $K_i = E(KS, i) for 1 \leq \ i \leq r$ These keys will be used by the return mixes to encrypt Bob's reply.

(3) The key seed (along with the number of hops r) is first encrypted with Alice's public key to form $y_0 = E_{\text{Alice}}(KS, r)$.

(4) Then, once for every mix on the return path, starting with the last, R_r, the following encryption is performed:

$y_i = (\mathcal{A}_{R_{r-i+1}}, E_{R_{r-i+1}}(K_{r-i+1}, y_{i-1}))$
(for $1 \leq i \leq r$)

The final outcome is:

$y_r = \mathcal{A}_{R_1}, E_{R_1}(K_1, \mathcal{A}_{R_2}, E_{R_2}(K_2, \ldots$
$\ldots E_{R_r}(K_r, \mathcal{A}_{\text{Alice}}, E_{\text{Alice}}(KS, r)) \ldots))$.

We refer to the resultant block, shown in Figure 5, as a *little onion*, similar in construction, but smaller than, the forward-path *onion*.

(5) Alice inserts the resulting RPI block into the beginning of the cleartext message she wishes to send. Then the procedure outlined in the previous section is followed until the last remailer on the forward path, F_f, is reached. F_f detects the

[8] Unless of course the message bears Alice's name or signature.

[9] Note that little can be done in case of the last hop mix.

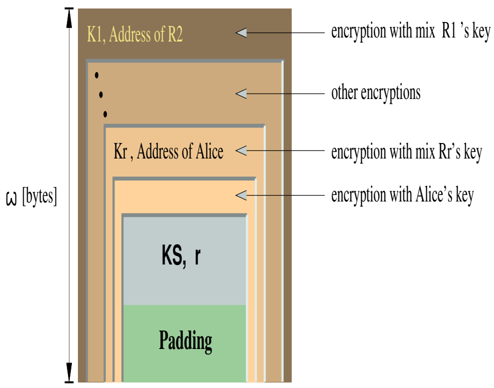

Figure 5: Return Path Information

RPI in the outbound message and modifies the mail header such that a later reply by Bob would be sent directly to R_1 and not to F_f.[10] (We assume that the RPI is "visible" to the last hop mix. It would be more secure to encrypt RPI for the destination – Bob – but we try to avoid requiring any cryptographic capability from Bob.)

5.2 Replying by recipient

As mentioned above, the message sent by Alice and received by Bob is prefixed with an RPI block. The RPI is meant to be treated *opaquely* by Bob.

Bob composes his reply as usual and simply prepends the RPI he received from Alice. He then sends his reply to the first mix on the return path, namely R_1.

The message received by R_1 is shown in Figure 6.

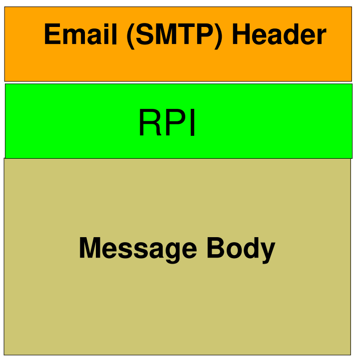

Figure 6: Bob's reply as received by R_1

5.3 Reply processing by remailers

Upon receiving Bob's reply, R_1 detects the included RPI and extracts it. Let us denote this original RPI by RPI_0. As the first mix on the return path, R_1, performs the following steps:

(1) Combine the header and body of the reply (without the RPI) into a string M'. This is the string that will ultimately reach Alice.

(2) Pad M' to size $\Omega - \omega$.

(3) Decrypt RPI_0, to reveal the random key K_1 and $\mathcal{A}_{R_2}$, the address of R_2. Let RPI_1 denote the new[11] RPI, which has one fewer layer of encryption.

(4) Encrypt $\lceil M' \rceil^{\Omega-\omega}$ with K_1 to form $Y_1 = K_1\{\lceil Header + Body \rceil^{\Omega-\omega}\}$.

(5) Send (RPI_1, Y_1) to R_2. Note that the size of this message is Ω.

The next $r-1$ remailers on the return path will perform a similar operation. At mix R_i:

(1) After reception of $(\mathrm{RPI}_{i-1}, Y_{i-1})$ decrypt RPI_{i-1} to reveal $\mathcal{A}_{i+1}$ and K_i. The resultant value is denoted RPI_i.

(2) Encrypt Y_{i-1} by K_i to form $Y_i = K_i\{Y_{i-1}\}$.

(3) Send (RPI_i, Y_i) to the next hop $\mathcal{A}_{R_{i+1}}$.

For the last mix on the return path, the operation is identical except that the next hop's address will be $\mathcal{A}_{\mathrm{Alice}}$ instead of $\mathcal{A}_{R_{r+1}}$.

It is important to note that a reply message is indistinguishable from a message on the forward path because both have size Ω. The structure of both messages look identical to an outside observer, i.e. encrypted gibberish.

A mix is able to determine whether a message belongs to the forward or reply flows by performing at most two decryption attempts.

If decryption of first ω bytes is successful then the message is on the reply path.

Otherwise, the message is on the forward path and the decryption of the entire message, Ω bytes, should be successful[12]

5.4 Handling replies at the originator

Eventually, Alice receives the string (RPI_r, Y_r) as Bob's reply. However, by this time, all layers of encryption have been removed except the last. Therefore Alice sees:

$$\begin{aligned} \mathrm{RPI}_r &= E_{\mathrm{Alice}}(KS, r) \quad \text{and} \\ Y_r &= K_r\{K_{r-1}\{\ldots K_1\{\lceil M' \rceil\}\ldots\}\} \end{aligned}$$

Decrypting RPI_r reveals KS and r and allows Alice to regenerate $K_1 \ldots K_r$. Successive decryptions of Y_r with these keys yield M'. We note that in Chaum's model [2], Alice has to remember the keys $K_1, K_2, \ldots, K_r$, in order to process the reply. In our

[10] In RFC-822-compatible systems, this is achieved by including a "Reply-To" field [6] in the header of the message sent to Bob.

[11] Assume for the time being that the size of this new RPI is still ω.

[12] If both encryption attempts fail the message is discarded.

scheme, keys are embedded in the reply, considerably simplifying the processing and allowing Alice to remain *stateless* with respect to outstanding messages.

Note that M' is composed of Bob's reply, and its header as seen by the first mix. This header can be used to identify Bob. Thus, a reply to an anonymous message is not equally anonymous.

5.5 Two-way Anonymous Conversation

Despite the above, it is possible, under some conditions, for Alice and Bob to communicate anonymously in both directions. Suppose that Alice begins by sending an anonymous message to a newgroup or a bulletin board. This message, among other things, includes an RPI. Since Bob does not know Alice, he does not trust her RPI but it represents the only way to communicate with Alice. He sends his reply M' to R_1 (1st hop in RPI) anonymously through mixes $X_1, X_2, \ldots, X_x$ (see Figure 7). In other words, Bob creates his own forward path and connects it to Alice's RPI.

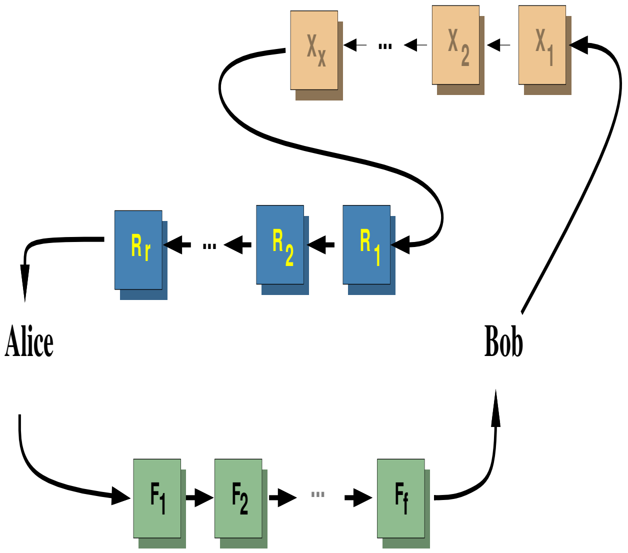

Figure 7: Bob's anonymous reply to Alice.

Bob can also include an RPI in his message so that Alice can reply to Bob's anonymous reply through yet another series of mixes, see Figure 8. *Thus, it is possible for two parties to communicate electronically in both directions without either party knowing the identity of the other.*

5.6 Security of replies

Unfortunately, it is difficult (if not impossible) to apply similar replay detection measures to replies as to forward-bound messages. This is because it is perfectly legitimate for multiple recipients to generate several responses to a single anonymous message. (this holds only if Alice explicitly allows replies by including an RPI block.)

Thus, Eve can mount a replay attack. Note, however, that in cases where replies are not wanted,

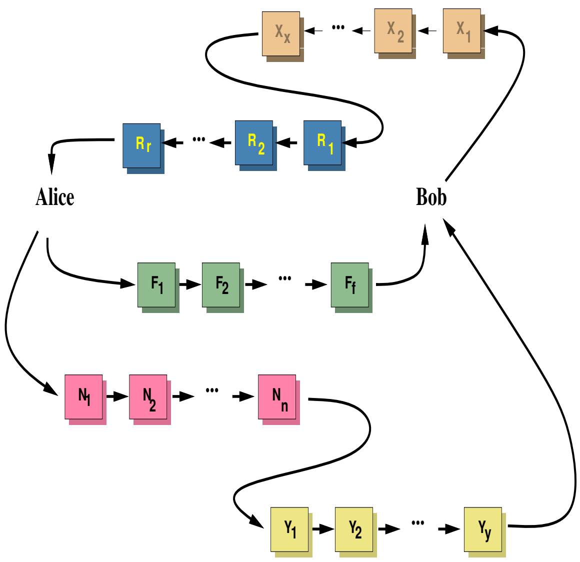

Figure 8: Alice replying to Bob's anonymous reply.

BABEL allows messages to be sent without an embedded RPI. Also, an RPI is not "tied" to a given sender; it is trivial to create an RPI with a fake return address. In other words, since RPI-s are not digitally signed, they can be repudiated.

5.6.1 Inter-Mix Detours

A simple yet powerful way of strengthening the security (i.e., untraceability) of replies by introducing inter-mix *detours*.

Let $R_1, R_2, \ldots, R_r$ denote the mixes on the return path. Normally a mix R_i $(0 \leq \; i \; < r)$ forwarded the reply to the next hop R_{i+1}. In the detour mode, R_i chooses a random forward path (called a *detour*) $D^i_1, D^i_2, \ldots, D^i_{r_i}$, which consists of normal mixes drawn from the global mix network. The message is then anonymously forwarded to R_{i+1} through these mixes as shown in Figure 9.

There is nothing special about detour-ed messages; they are a regular anonymous messages only constructed by a mix and not a user.

A detour ensures that a message leaving R_i appears different for each reply, in particular, during a replay attack. Compare this to the previous case where replies to the same anonymous message can be correlated by merely examining the exposed RPI.

Messages on the forward path could also be detoured. However, if the mixes on the deviated path further detoured messages, endless detour loops would occur. To avoid this problem, detoured messages would have to be tagged accordingly. An important benefit that can be derived from detouring forward-bound messages is that, unlike Chaum's mixes [2], *we can guarantee that even the originator of an anonymous message can not recognize its own message as it leaves a mix.*

One slight drawback of introducing inter-mix de-

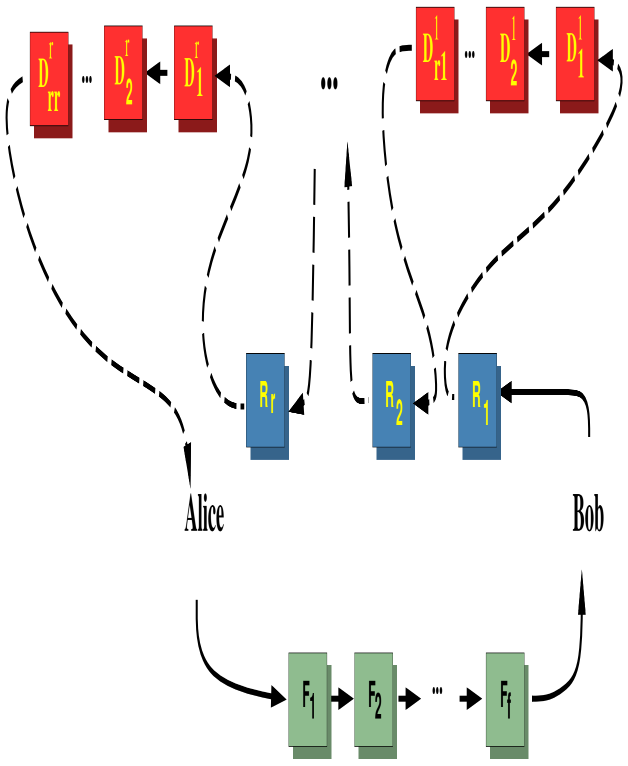

Figure 9: Inter-mix detours on replies.

tours is that a mix now has to know about other mixes; thus far, it has not been a requirement.

5.6.2 Indirect replies

An entirely different approach to replies can also be envisaged: instead of delivering a reply directly to Alice, Bob can deliver it to a local newsgroup with a special number tag. Alice scans this newsgroup for replies matching that number tag. This method is roughly analogous to the broadcast solution as described in [10].

6 Keeping Message Sizes Constant

In principle, a cryptosystem where encrypted messages are of the same length as the corresponding cleartext can be devised, e.g. CFB mode of DES with a pre-distributed initialization vector. In practice, however, ciphertext is usually somewhat longer than cleartext. In hybrid-key cryptosystems the size increase is particularly noticeable due to the need to include an encrypted random session key in addition to the ciphertext. Conversely, decryption results in a shorter message.

Thus, the length of an email message would decrease after each decryption as it travels through the mixes. The differences in size can be exploited by Eve.

The problem can be solved if each mix pads the outgoing message to Ω. Although all messages would have the same size for an eavesdropper, the decrease would still visible to remailers. This allows them to make educated guesses as to the number of preceding or following hops, and is contrary to one of our goals set in Section 2.2.

Each mix should know only the identity of the previous and next hop and nothing else about the path of a message. The first and last hops are a little different because they can learn the identity of the sender and the recipient, respectively.

Furthermore, the number of preceding and following hops should be kept secret. Although the message sent by Alice is indistinguishable from other inter-mix traffic, the first hop can infer that it is the first hop by comparing Alice's address with the list of known mixes. In a similar fashion, the last hop can deduce that it is the last. However, all others, i.e. intermediate hops, should not know the number of preceding hops nor the number of following ones.

Chaum [2] presents a general solution where data is divided into a fixed number of fixed-size blocks. This is the solution implemented in the Mixmaster package[7].

Here we present another approach that is simpler and more storage-efficient. The basic idea is to ensure that some padding (encrypted or not) always follows information-carrying data. An example should make the point clear.

Let string C of length Ω be composed of M bytes of data followed by $P = \Omega - M$ bytes of padding. Also suppose the encrypted version of C is denoted by C' having length $\Omega + \delta$. If $\delta < P$ then trimming δ trailing bytes of C' has no impact on the encrypted version of the data but only on the encrypted version of the padding, see Figure 10. In other words, trimming δ bytes results merely in the loss of the original padding but not in data loss.

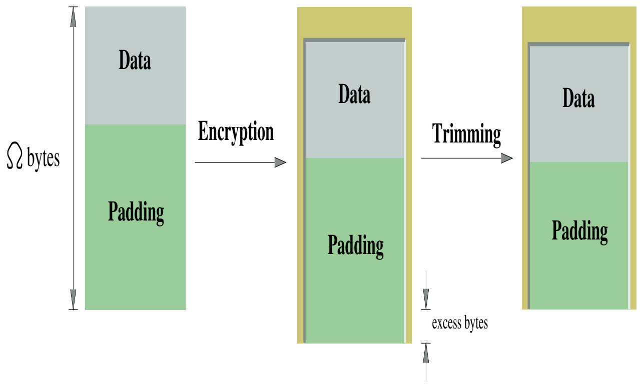

Figure 10: Padding—Encrypting—Trimming

For the previous statement to hold, the encryption algorithm should be such that correct decryption of a given block depends on some or all of the previous blocks but not on following blocks. This is true for most encryption algorithms. We also note that if the encryption package used embeds CRC or length information about the cleartext, alterations made to the ciphertext will be detected, leading to possible rejection of the message. This issue is further discussed in Section 8.2.2.

7 Heeding anonymity

In the preceding sections we defined the notion of a mix, the potential threats facing it and requirements for constructing mixes that provide bi-directional anonymity. This section attempts to formalize and analyze the degree of anonymity a particular remailer system provides. In particular, the notions of confusion and staunchness are introduced and defined.

7.1 Fixed-Path Systems

Until now we made an assumption that a mix path is chosen at random from a large pool of available mixes. This should not necessarily be so. An interesting way to increase the overall traffic load is to use the same fixed mix path for all messages [24]. We denote this path by $M_1, M_2, \ldots, M_m$. In this configuration, messages always enter the system at M_1, are forwarded to M_2, then to the next mix, and so on until they leave the system at M_m.

By forcing all messages to visit all mixes pertaining to the fixed path, the traffic going through each is maximal. There are other advantages of using a fixed path. The mix network becomes more reliable, less chaotic and much easier to manage.

Maximizing traffic load might seem contrary to good engineering practices. Clearly, if a mix is overwhelmed by sheer traffic volume, data loss can occur. This is not a serious drawback because, as the traffic increases beyond the processing capacity of the mixes, other fixed paths can be introduced to offload the previous fixed path(s).

Owing to practical considerations, the number of mixes on the fixed path, m, is clearly limited. Thus, the advantage of using a large number of mixes is lost. Now it is much easier for Eve to monitor the entire system. She can even learn a great deal by watching only the first and last mixes, M_1 and M_m. Consider the following attack where Eve allows only a single message to trickle into a interval batching mix network.

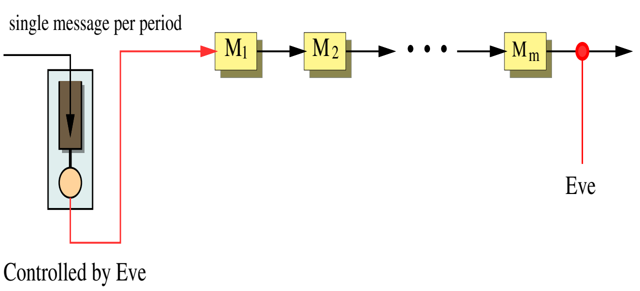

Figure 11: The Trickle Attack

This attack is referred to as the trickle attack because Eve allows only a single genuine message to enter the system. By observing the output of the last mix, M_m, she can correctly correlate the genuine message with its corresponding output.

Decoys might be used to outfox the trickle attack. However, as many users would be alarmed or even upset by receiving decoy messages, we do not allow them to leave the mix network. Unfortunately, inter-mix decoys do not confuse Eve.

7.2 System staunchness, miss & guess factors

We define the *miss factor*, denoted $\mathcal{M}$, for a mix network as the probability of making an incorrect correlation between a message entering the mix network and a message leaving it. It represents the measure of confusion introduced by the mixes. Similarly, the *guess factor*, denoted $\mathcal{G}$, is defined as the probability of making a correct correlation. (Obviously, $\mathcal{G}+\mathcal{M}=1$.)

Consider the fixed-path case where the intervening mixes use regular batching with the batch size set to N. Then, $\mathcal{G}$ for the fixed path is equal to $1/N$. It is interesting to note that the result is identical to the guess factor of a single mix. What is then the advantage of chaining through several mixes?

A chain of mixes is more *secure* than a single mix because Eve has to subvert all mixes in order to break the anonymity chain. In other words, a chain of mixes is more secure than a single mix but not necessarily more confusing. We define the *staunchness*, $\mathcal{S}$, of a mix network as the number of secret keys needed to defeat message anonymity. In all schemes described this far, staunchness is equal to the number of mixes a message travels through.

7.3 The Quest for Confusion

Consider the fixed path case where intervening mixes use interval-based batching instead of regular batching. Assuming the clocks of remailers are perfectly synchronized and message transmission time is small but non-zero[13], the itinerary of a group of messages arriving during interval i is depicted in Figure 12.

A message entering the system at interval i will leave it at time $(m \cdot T)$, along with the rest of the messages entered during the same interval. The guess factor for period i is given by

$$\mathcal{G}_i = \frac{1}{n_i},$$

where n_i is the number of messages entering the system in interval i.

Thus, if few messages enter the system, the probability for correct correlation is close to one. This is what one would expect by intuition.

For obvious reasons, the higher the value of the miss factor, the better. One could simply increase the duration of the interval, T, to augment the average number of incoming messages per period. However, this has a negative impact on the average delay experienced by messages. They will be delayed on the average by $T/2$ in the first mix and for a full period at the following mixes. Thus, the total average delay for

[13] So that messages arrive at the following mix during a new interval.

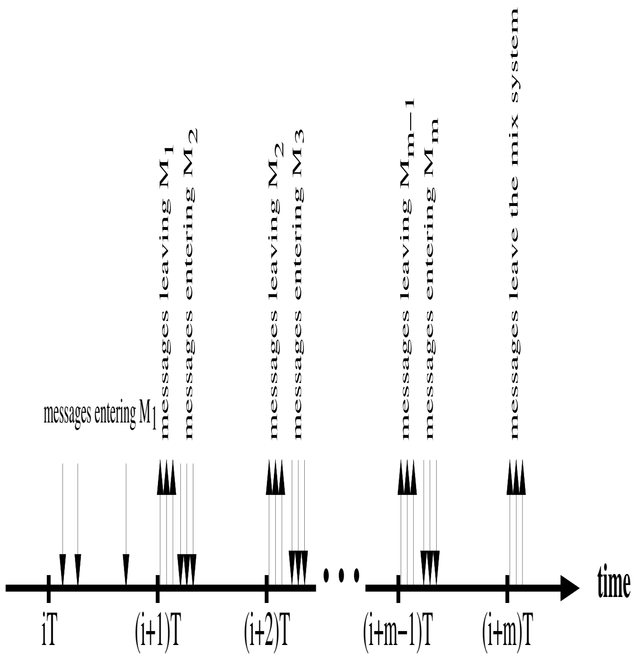

Figure 12: Interval batching with synchronized clocks

the fixed path, neglecting transmission and processing time, is given by

$$E[Delay] \quad = \quad T(\frac{1}{2} + m - 1) \quad \text{[sec]}$$

where m is the number of remailers on the fixed path.

7.3.1 Probabilistic deferment

Continuing our pursuit of confusion, we now introduce a new scheme based on the time interval method but with an added twist. The "twist" is that at the end of each time interval, some of the *incoming* messages are deferred for an additional time period while all other messages are sent with no further delay[14]. We refer to this scheme as probabilistic deferment with interval batching.

The decision to defer a given incoming message is taken by flipping a biased coin. Let q be the probability of forwarding the message at the end of the current interval and $d = 1 - q$ the probability of deferring it for an additional period.

Let the random variable K denote the number of times a given message leaving the mix system has been deferred. The probability mass function of K is given by

$$\mathrm{P}\{K = k\} \quad = \quad \binom{m}{k} q^{m-k} d^k \quad \text{where } k = 0, \ldots, m,$$

[14] Incoming and deferred messages are distinguished by keeping appropriate state information.

which is the binomial distribution. The expected value of K is simply

$$E[K] \quad = \quad md$$

Thus, with the new scheme, a message on the average will be delayed by:

$$E[\text{delay}] \quad = \quad T(\frac{1}{2} + m - 1) + \underbrace{T * m * d}_{\text{avg addtl delay}} \quad \text{[sec]}$$

Note that in the worst case a message may be delayed as long as $2Tm$ seconds; delayed for a full interval and also deferred on all m mixes.

With the new scheduling policy the opponent has to guess both the interval to which a message belongs (i.e k) and also its position in that interval. Presuming that the number of messages arriving at each period is roughly the same[15], Eve's best guess is to assume the most likely deferment event.

Thus the guess factor for the new policy for the interval i, designated $\widehat{\mathcal{G}}_i$, is given by the guess factor for simple interval batching, times the probability of the most likely deferment event, i.e.

$$\widehat{\mathcal{G}}_i \quad = \quad \mathcal{G}_i \cdot \mathrm{P}\{\text{most likely } k\}$$

For a binomial variable B, with parameters (m, d), where $0 < d < 1$, as b goes from 0 to m, $\mathrm{P}\{B = b\}$ first increases monotonically and then decreases monotonically, reaching its largest value for[16] $\lceil \mathrm{E}[B] \rceil$, the smallest integer greater than or equal to $m \cdot d$. For a rigorous proof, refer to [34]. For a less rigorous but amusing proof, the reader can approximate the binomial by the Poisson distribution, generalize the factorial to the gamma function[17] and then take the derivative with respect to a now continuous b.

Figure 13 shows the probability of the most likely event, $\mathrm{P}\{B = \lceil \mathrm{E}[b] \rceil\}$, as a function of the deferment probability d for even and odd values of m.

Clearly, for odd values of m, the probability of the most likely event is minimal for $d = 1/2$. This is a little different for even values of m, for which the minimum is reached for values of d not too far away from $1/2$.

For a numeric example, suppose $m = 5$ and $d = q = 1/2$. The most likely value for k is 3, with probability $\frac{10}{32}$. Thus, the probabilistic deferment method introduces an additional uncertainty of $\frac{10}{32}$. If we had simply doubled the time interval to $T' = 2 \cdot T$, as to have the same delay in the worst case, then the decrease in the guess factor would be only $1/2$. The probabilistic deferment method compares well with simple interval batching for all values of $m > 1$, even in the worst case.

[15] Otherwise, messages are likely to belong to the most populated interval.

[16] "E" does not mean encryption here.

[17] Like the exponential function the gamma function is also equal to itself when derived. However, it is only defined for $\Re^+$.

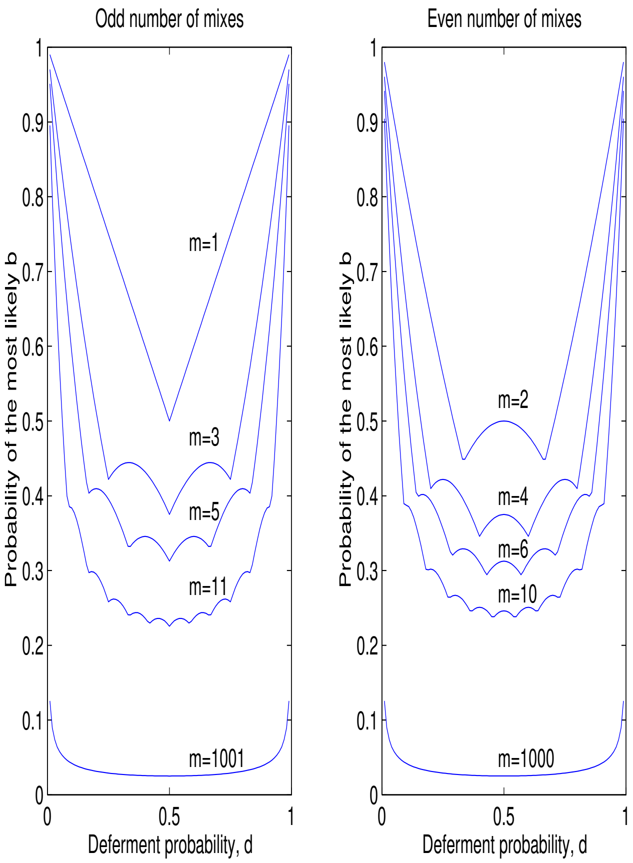

Figure 13: Probability of the most likely b as a function of d

7.3.2 A Hybrid Approach

A hybrid configuration, referred to as *fixed-set random order path*, imposes a fixed set of mixes but allows traversing them in any order chosen at random, with each mix visited only once. As with the fixed path method, the traffic load is optimal. However, there are no critical lines. Eve must observe and control all communications lines to defeat the mix network. The probabilistic deferment approach can also be put to use to increase the confusion factor further.

It is very difficult to calculate the confusion factor for the fixed-set random order system. However, it combines some of the best features of the methods mentioned so far.

8 Implementation

An anonymous remailer conforming to the ideas and requirements described in this paper has been implemented the IBM Zurich Research Laboratory during first half of 1995. This section discusses some of the salient aspects of the implementation.

8.1 Computing environment

The popular script language Perl [32, 33] was used to implement BABEL. Perl is readily available on most Unix platforms and is well-suited for processing loosely structured data such as email messaged. We opted for the latest incarnation of Perl, version 5.

We recognize that there is an inherent performance cost involved in using an interpreted language such as Perl. However, the impact of interpreting the code at run time is negligible compared to that of cryptographic operations, which are notoriously costly in terms of processing power.

8.2 Pretty Good Privacy or PGP

It being the most popular email encryption software, we chose PGP to provide the cryptographic base. PGP combines the convenience and security of public-key algorithms with the high speed of conventional cryptography. It offers full-blown message privacy and authentication, based on RSA [30] and IDEA [14, 15].

Since it was designed with the mass appeal in mind, PGP is well-suited for interactive use. Unfortunately, this is not the case for automated (batch) processing; error conditions require unexpected user interaction, and the return codes are at times confusing.

8.2.1 PGP file format

With PGP, an email message can be compressed, encrypted and signed, but the user can view its contents and verify its signature with a single command. At the byte level this is achieved by embedding a compressed packet inside a hybrid RSA-IDEA encrypted packet. This packet itself is then embedded in a signature packet (Figure 14) which can in turn be embedded in a radix-64 ASCII armor.

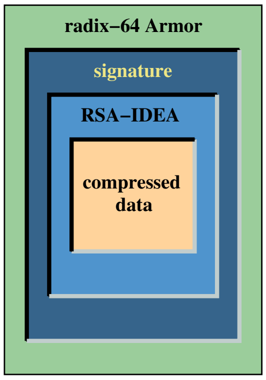

Figure 14: Multiple Packet Embedding

PGP recursively processes each packet type until an unknown type, i.e. user data, is encountered. Although this might be the correct behavior at the user level, it is inadequate when multiple encryption is used. In that case, PGP[18] attempts to continue decrypting after a first successful decryption. The second decryption operation will usually fail because the secret key needed to perform the operation will be missing (a mix does not know the secret key of other mixes).

[18] Behavior observed with the "+force" option required for in batch processing.

Furthermore, PGP is meant to be used for email privacy and authentication but not sender anonymity. PEM is even worse in this respect, as the unencrypted PEM message headers contain identification of both *sender* and recipient [31]. The cleartext part of PGP message headers also contains sensitive information that can be used by an attacker to correlate messages. This potential threat was carefully studied, and a version-independent PGP format parser was developed at the earlier stages of the project.

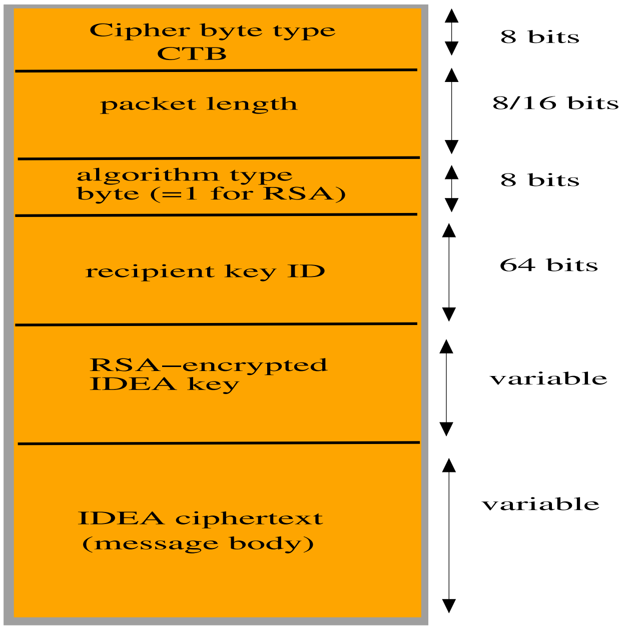

Figure 15: Data format for encrypted file

8.2.2 Side effects of encryption

By default, PGP attempts to compress cleartext before encrypting it[19]. However, since uniform message size is a concern, compression is always turned off. This prevents messages from shrinking.

There is another reason for turning off compression. In compression mode, PGP adds a CRC of the cleartext into the ciphertext. This causes PGP to reject files altered in any way, particularly trimmed files. As mentioned in Section 6, trimming is used to enforce uniform message size. Fortunately, when compression is turned off, PGP records only the length of the cleartext message. Thus, alterations to data length are detected but not those to contents.

To be precise, PGP rejects messages that are shorter than the prerecorded value, but accepts longer ones. This behavior can be explained by considering that, in an email message that includes PGP ciphertext, the cleartext (e.g., mail headers) usually precedes the PGP part of the message. Moreover, additional cleartext (e.g. a cleartext signature) usually follows the PGP ciphertext. We capitalize on this behavior to implement the forward and reply messages indistinguishably, as presented in Section 5.3.

[19] Some think that compression enhances security; we do not.

8.2.3 Radix-64 format

As PGP-encrypted files are in binary format, some sort of conversion must take place to send encrypted data over 7-bit channels such as email. A remarkably simple and efficient conversion method is radix-64 armoring. It is defined in [16].

8.3 Remailer deployment

A BABEL mix is designed to act as a filter installed in the *.forward* file. Refer to [5] for further information on email filters. Any user can transform his computer account into an anonymous remailer in a matter of minutes, without having any administrator privileges. Personal email is treated as usual, but anonymous mail is filtered and processed without ever cluttering the user's mailbox[20]. This is compatible with the Internet's *populist* philosophy. However, note that this paves the way for a security breach. Since BABEL is designed with a minimum of human intervention in mind, the password needed to access the secret key of a remailer is stored in cleartext, in a read-protected configuration file. Although this file is not accessible to a casual user, the system administrator can usually override the safeguards. Furthermore, a popular remailer site can attract swarms of messages. This can result in serious performance degradation on the local host.

The actual deployment of BABEL has been delayed due to U.S. export restrictions on cryptographic paraphernalia. Restrictions apply not only to cryptography per se but also to equipment that makes use of cryptography. In particular, although BABEL does not contain a single line of cryptographic code and relies completely on PGP it is still subject to the aforementioned export restrictions.

8.4 Proxies

In order to appeal to the greatest number of users, BABEL offers a so-called *proxy mode* of operation. In this mode, a user with no BABEL software can ask any mix to compose and forward an anonymous message on the user's behalf. The proxy mix is also able to substitute itself for the user in order to process multiply encrypted replies. Consequently, it is possible for any bare-bones email user to send anonymous messages and receive replies.[21]

The proxy mode of operations is somewhat less secure because traffic to the proxy mix flows in cleartext. However, users equipped PGP but no BABEL software can send their orders encrypted with the proxy mix's public key.

8.5 Message length—Concrete values

The Internet email "bible", RFC 1123 [13], specifies that any mailer software should be able to send and receive messages at least 64 Kbytes in length (including header). Taking into account a 33% increase of radix-64 armoring, the maximum uniform message

[20] Unless an error occurs while processing the message.

[21] This is particularly applicable to non-Unix users.

size, Ω, we could safely adopt is 48 Kbytes[22]. Being concerned by network bandwidth, we opted for half that number, i.e. 24 Kbytes.

For a 512-bit public key, PGP increases message size by about 115 bytes at each encryption. Experiments show that the thickness of a layer of the anonymous onion is on average approximately 220 bytes. Therefore, when 2 Kbytes of padding are used, a message can safely include nine layers of encryption. The recommended RPI size ω is 1.5 Kbytes. This allows approximately seven mixes on the return path.

We intentionally chose not to provide support for larger files. This is the accepted practice on existing remailers. It is meant to frustrate the anonymous transmission of graphic files, which tend to be very large[23]. It is still possible to split larger files into smaller pieces and send them anonymously.

8.6 Time Synchronization & Replay Detection

As mentioned in Section 3.2.2, each layer of the onion created by Alice includes a time stamp. The value of the time stamp, referred to as Θ, is the number of seconds elapsed in seconds since January 1, 1970 GMT, to the moment of message composition by the sender.

A BABEL mix uses a two-step replay detection. First, it records a unique identifier of the message as described in Section 3.2.2. As long as the record is in the database, replays are detected. However, in order to keep the database size reasonable, the record is deleted at time $(\Theta + \Delta)$. Thereafter, any message bearing the timestamp Θ or older will be discarded as being too old; not necessarily for being a replay.

Time stamps are introduced merely to keep the replay database small. Thus, only loose clock synchronization is needed. Assuming the *total* delay experienced by messages at remailers to be about one hour, we chose Δ to be 24 hours, one order of magnitude larger than message the delay. Thus, the time it takes to visit all mixes on the forward path is considered negligible with respect to Δ.

With such a coarse value of Δ, it is sufficient that hosts keep clocks accurate within a day for the system to function properly.

9 Conclusions

This paper presented an anonymous remailer system called BABEL . BABEL is flexible enough to allow both sending and receiving anonymous electronic messages. Anonymity criteria have been defined in order to compare degrees of anonymity provided by various configurations.

The basic components of BABEL, mixes, are not aware of each other and learn very little about messages they process. In contrast to some currently-operating remailers, BABEL mixes do not depend on (potentially treacherous) alias tables.

The software implementation of BABEL is based on freely available ingredients: *Perl* and PGP. At the same time, the system remains accessible to users with only a basic email capability through the use of its proxy mode.

A BABEL mix can be very easily set up by any user having only a simple Unix account. However, it is envisaged that setting up an Internet-wide mix network (mesh) will take some time.

As with any new technology, some abuse is unavoidable. *Caveat Emptor!*

Acknowledgments

The authors wish to thank Ph. Janson, M. Waidner, M. Steiner, R. Hauser and the *anonymized* referees for many helpful comments and suggestions.

References

[1] D. Akst, "Postcard from cyberspace," *Los Angeles Times*, February 22 1995.

[2] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, v. 24, n. 2, Feb 1981, pp. 84–88.

[3] D. Chaum, "Security without Identification: Transaction Systems to make Big Brother Obsolete," *Communications of the ACM*, 28/10, 1985, pp. 1030–1044.

[4] D. Chaum, "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability," *Journal of Cryptology*, 1/1, 1988, pp. 65–75.

[5] B. Costales, E. Allman and N. Rickert, "Sendmail", O'Reilly & Associates, 1993.

[6] D. H. Crocker, "Standard for the format of Arpa Internet messages", RFC 822, August 1982.

[7] L. Cottrell, "Mixmaster and Remailer Attacks," http://obscura.com/~loki/remailer-essay.html.

[8] D. W. Davies and W. L. Price, "Security for Computer Networks," John Wiley & Sons, 1984, pp. 137–143 .

[9] D. E. R. Denning, "Cryptography and Data Security," Addison-Wesley, 1982

[10] D. J. Farber and K. C. Larson, "Network Security via Dynamic Process Renaming," *Fourth Data Communications Symposium*, Oct 1975, Quebec City, pp. 8–18.

[11] J. Helsingius : Press release, February 20th 1995.

[12] E. Hughes, "Cypherpunks Manifesto," distributed on Usenet and various mailing lists, March 1993.

[13] Internet Engineering Task Force, "Requirements for Internet Hosts – Application and Support," RFC 1123, October 1989.

[14] X. Lai, "On the Design and Security of of Block Ciphers," ETH Series in Information Processing , v. 1, Konstanz: Hartung-Gorre Verlag, 1992.

[15] X. Lai and J. Massey, "A proposal for a New Block Encryption Standard," *Advances in Cryptology—EUROCRYPT '90 Proceedings*, Berlin: Springer-Verlag, 1991, pp. 389–404.

[22] not taking the header size into account.

[23] and of questionable nature.

[16] J. Linn, "Privacy Enhancement for Internet Electronic Mail — Part I: Message Encryption and Authentication Procedures," RFC 1421, Feb 1993.

[17] S. Maguire, "Writing Solid Code.", Microsoft Press, 1993, pp. 179–180.

[18] T. May, "Crypto Anarchy and Virtual communities," *Internet Security Journal*, April 1995.

[19] D. L. Mills, "Algorithms for synchronizing network clocks." RFC 956, September 1985.

[20] D. L. Mills, "Experiments in network clock synchronization," RFC 957, September 1985.

[21] D. L. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," RFC 1305, March 1992.

[22] W. Mossberg, "Personal technology," *Wall Street Journal*, Jan. 26 1995.

[23] J. Quittner, "Unmasked on the Net," *Time magazine*, March 6 1995.

[24] A. Pfitzmann , "How to implement ISDNs without user observability—Some remarks," Institut für Informatik, University of Karlsruhe, Interner Bericht 14/85, 1985.

[25] A. Pfitzmann, B. Pfitzmann and M. Waidner, "ISDN-Mixes: Untraceable Communication with Very Small Bandwidth Overhead," *GI/ITG Conference: Communication in Distributed Systems*, Mannheim Feb. 20–22 1991, Informatik-Fachberichte 267, Springer-Verlag, Heildelberg 1991, pp. 451–463.

[26] A. Pfitzmann and B. Pfitzmann, "How to break the direct RSA-implementation of MIXes," *Advances in Cryptology—EUROCRYPT '89 Proceedings*, Berlin: Springer-Verlag, 1990, pp. 373–381.

[27] A. Pfitzmann and M. Waidner, "Networks Without User Observability—design options," *Eurocrypt 85*, Springer-Verlag, Berlin 1986, pp. 245–253. Revision in: *Computers & Security*, 6/2 1987, pp. 158–166.

[28] J. B. Postel, "Simple Mail Transfer Protocol", RFC 821, August 1982.

[29] R. Rivest and A. Shamir, "How to Expose an Eavesdropper," *Communications of the ACM*, v.21, n. 2, Feb 1978, pp. 120–126.

[30] R. Rivest, A. Shamir, and L. M. Adleman, "Cryptographic Communications System and Method," U.S Patent 4,405,829, 20 Sep 1983.

[31] B. Schneier, "Applied Cryptography," John Wiley & Sons, 1994.

[32] R. Schwartz, "Learning Perl," O'Reilly & Associates, 1993

[33] L. Wall and R. Schwartz, "Programming Perl," O'Reilly & Associates, 1993.

[34] R. Sheldon, "A first Course in Probability," Macmillan, fourth edition, 1994, pp. 147–167.

[35] P. Zimmerman, "PGP User's Guide", included in PGP distribution 2.6i, October 1994.

[36] P. Zimmerman, "PGP 2.6 file formats", included in PGP distribution 2.6, May 1994.