

Comparison between two practical mix designs

Claudia Díaz Len Sassaman Evelyne Dewitte

Abstract

We evaluate the anonymity provided by two popular email mix implementations, Mixmaster and Reliable, and compare their effectiveness through the use of simulations which model the algorithms used by these mixing applications. In order to draw accurate conclusions about the operation of these mixes, we use as our input to these simulations actual traffic data obtained from a public anonymous remailer (mix node). We determine that assumptions made in previous literature about the distribution of mix input traffic are incorrect, and our analysis of the input traffic shows that it follows no known distribution. We establish for the first time that a lower bound exists on the anonymity of Mixmaster, and discover that under certain circumstances the algorithm used by Reliable provides no anonymity. We find that the upper bound on anonymity provided by Mixmaster is slightly higher than that provided by Reliable. We identify flaws in the software code in Reliable that further compromise its ability to provide anonymity, and review key areas which are necessary for the security of a mix in addition to a sound algorithm. Our analysis can be used to evaluate under which circumstances the two mixing algorithms should be utilized to best achieve anonymity and satisfy their purpose. Our work can also be used as a framework for establishing a security review process for mix node deployments.

1 Introduction

The Internet was initially perceived as a rather anonymous environment. Nowadays, we know that it can be a powerful surveillance tool: anyone willing to listen to the communication links can spy on Internet users, and search engines and data mining techniques are becoming increasingly powerful.

Privacy does not only mean confidentiality of the information; it also means not revealing information about who is communicating with whom. Anonymous remailers (also called *mixes*) allow their users to send emails without disclosing the identity of the recipient to a third party. They also allow the sender of a message to stay anonymous to the recipient.

The objective of this work is to have quantitative results on the anonymity actually provided by two mix software implementations in wide deployment, in order to test the actual anonymity provided to the users of the remailer service, and to compare the two different designs. We evaluate anonymity in a single-node context. In order to assess the anonymity provided by the entire remailer

network, additional considerations are necessary. As individual nodes are the basic component to the network of mixes, we aim to provide information to be considered when choosing this component. We have used as input real-life data gathered from a popular remailer, and simulated the behaviour of the mix.

2 Mixes

Mixes are the essential building block to provide anonymous email services. A mix is a router that hides the relationship between incoming and outgoing messages. The mix changes the appearance and the flow of the messages. In order to change the appearance of the messages the mix uses some techniques, such as padding and encryption, thus providing bitwise unlinkability between inputs and outputs. Techniques like reordering and delaying messages, and generating dummy traffic are used to modify the flow of messages. This modification of the traffic flow is needed to prevent timing attacks that could disclose the relationship between an input and an output messages by looking at the time the message arrived to and left from the mix.

The idea of mixes was introduced by Chaum [Cha81]. This first design was a *threshold mix*, a mix that collects a certain number of messages and then flushes them. Since then, variants on this first design have been proposed in the literature. In this paper, we focus on two practical mix designs that have been implemented and are part of the Mixmaster remailer network[Cot95], which has been providing anonymous email services since 1995.

The first design is called “Mixmaster” (as the remailer network) because it is descended from the original software program designed by Cottrell [Cot, MCPS03]. The second design, called “Reliable”, uses a different reordering strategy. [RP99] The details of the two remailers are explained in the following sections. We compare version 3.0 of the Mixmaster software and version 1.0.5 of Reliable.

2.1 Mixmaster

Mixmaster¹ is a *pool* mix. Pool mixes process the messages in batches. They collect messages for some time, place them in the pool (memory of the mix), and select some of them for flushing in random order when the flushing condition is fulfilled. Mixmaster is a timed mix which has a *timeout* of 15 minutes. During this period of time, it collects messages that are placed in the pool of the mix. When the timeout expires, the mix takes a number of messages from the pool that are forwarded to their next destination, which may be another mix or a final recipient. The algorithm that determines the number s of messages that

¹Mixmaster version 3.0, as well as Reliable, also optionally support the older “Cypherpunk” remailer message format. For the purposes of this paper, we are assuming that the remailers are being operated without this support. As anonymity sets for the two protocols generally do not overlap, this does not impact our results. The Cypherpunk remailer protocol is known to contain numerous flaws, and should not be used if strong anonymity is required[Cot, DDM03].

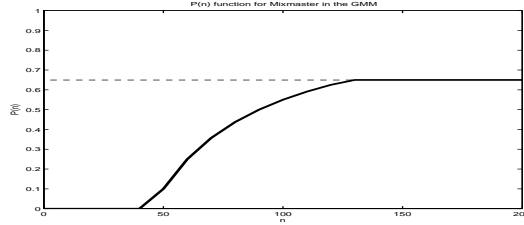


Figure 1: Mixmaster in the GMM

are sent in a *round* (one cycle of the mix) is a function of the number n of messages in the pool:

```

if (n<45) s=0;
else if (0.35*n < 45) s=n-45;
else s=0.65*n;

```

Mixmaster is represented in the generalised mix model proposed by Díaz and Serjantov [DS03b] as shown in figure 1. In this model, the mix is represented at the time of flushing. The function $P(n)$ represents the probability of a message of being flushed by the mix, as a function of the number n of messages in the pool. Note that $P(n) = s/n$.

2.2 Reliable

Reliable is based on the Stop-and-Go (*SG Mix*) mix proposed by Kesdogan *et al.* in [KEB98]. In SG mixes (also called *continuous mixes*), the users generate a random delay from an exponential distribution. The mix holds the message for the specified delay and then forwards it. The messages are reordered by the randomness of the delay distribution. This mix sends messages continuously: every time a message has been kept for the delay time it is sent by the mix.

Reliable interoperates with Mixmaster on the protocol level by using the Mixmaster message format for packet transfer. Reliable uses a variant of the *S-G Mix* design. In Reliable, the delay may be chosen by the user from an exponential distribution of mean one hour. If the user does not provide any delay to the mix, then the mix itself picks a delay from a uniform distribution, being the maximum and minimum of the uniform one and four hours, respectively. Note that these parameters of the delay distributions are configurable, and therefore many remailer operators may set them lower in order to provide a faster service.

2.3 Dummy traffic

A dummy message is a *fake* message introduced in the mix network in order to make it more difficult for an attacker to deploy attacks that can compromise

the anonymity of a message. The dummy message is normally produced by the mixes, and they select as destination another mix instead of a real recipient.

Dummies are indistinguishable from real messages as they travel in the mix network. As they are introduced to prevent traffic analysis, the dummy policy should maximize the number of possible destinations for the messages flushed by the mix. Dummy traffic has an impact when analyzing the mix network as a whole. We have made measurements that show that the impact of dummies on the anonymity provided by a single mix is very small. In order to make the fair comparison of Mixmaster and Reliable easier, we have not taken into account the dummy policies of these two mixes in the results presented in this paper.

Dummy policy of Mixmaster Every time a message is received by Mixmaster, an algorithm runs to generate d_1 dummies that are inserted in the pool of the mix. The number d_1 of dummies generated follow a geometrical distribution whose parameter has the default value of $1/10$. Moreover, every time Mixmaster flushes messages, it generated a number d_2 of dummies that are sent along with the messages. The number d_2 of dummies follows a geometrical distribution whose parameter has the default value $1/30$.

Dummy policy of Reliable Reliable’s default dummy policy consists in generation 25 dummies every 6 hours. The time these dummies are kept in the mix is generated from a uniform distribution whose minimum value is 0 and maximum is 6 hours.

3 Anonymity metrics

In this section we introduce the anonymity metrics for mixes. We remark on the particularities of some mix designs (binomial mixes and threshold mixes). Also, we present the attack model which we have considered.

Anonymity was defined by Pfitzmann and Köhntopp [PK00] as “*the state of being not identifiable within a set of subjects, the anonymity set*”.

The use of the information theoretical concept of entropy as a metric for anonymity was simultaneously proposed by Serjantov and Danezis in [SD02] and by Díaz *et al.* in [DSCP02]. The difference between the two models for measuring anonymity is that in [DSCP02] the entropy is normalized with respect to the number of users. In this paper we will use the non-normalized flavour of the metric.

The anonymity provided by a mix can be computed for the incoming or for the outgoing messages. We call this *sender anonymity* and *recipient anonymity*.

Sender anonymity. In order to compute the sender anonymity, we want to know the effective size of the anonymity set of senders for a message output by the mix. Therefore, we compute the entropy of the probability distribution that relates an outgoing message of the mix (the one for which we want to know the anonymity set size) with all the possible inputs.

Recipient anonymity. If we want to compute the effective recipient anonymity set size of an incoming message that goes through the mix, we have to compute the entropy of the probability distribution that relates the chosen input with all possible outputs.

Note that in these two cases, the metric computes the anonymity of a *particular* input or output message, it does not give a general value for a mix design and it is dependent on the traffic pattern. The advantage of this property is that mixes may offer information about the *current* anonymity they are providing. The disadvantage is that it becomes very difficult to compare theoretically different mix designs. Nevertheless, it is possible to measure on real systems (or simulations) the anonymity obtained for a large number of messages and provide comparative statistics, as we do in this paper.

In order to measure Mixmaster’s sender and recipient anonymity, we have applied the formulas provided by Díaz and Preneel in [DP04]. The anonymity of Reliable has been measured using the formulas presented in Appendix A. Note that we could not apply the method used by Kesdogan [KEB98] because we did not make any assumption on the distribution of the mix’s incoming traffic (Kesdogan assumes incoming Poisson traffic).

3.1 Attack model

The anonymity metric computes the uncertainty about the sender or the recipient of a message, given that some information is available. In our case, we assume that the mix is observed by a passive attacker, who can see the incoming and outgoing messages of the mix. The attacker knows all internal parameters of the mix so he can effectively compute the anonymity set size for every incoming and outgoing message.

4 Simulators

We have implemented Java simulators for Reliable and Mixmaster. We have fed the simulated mixes with real input, obtained by logging a timestamp every time a message arrived to a working Mixmaster node (note that the information we logged does not threaten the anonymity of the users of the mix). We have used four months of incoming traffic (July-November 2003) in order to obtain the results presented in Section 5.

In order to make a fair comparison, we have set the mean of the exponential delay of Reliable (default 1 hour) to be the same as provided by Mixmaster for the given four months of input (43 minutes). We have assumed users choose their delays from an exponential distribution. The mix-chosen uniform delay option has not been taken into account, due to the unfeasibility of implementing algorithms that compute the anonymity for such a delay distribution without making assumptions on the traffic pattern, as explained in Appendix A. Moreover, the choice of a uniform delay for the messages is completely non-standard.

The simulators log the delay and the anonymity for every message. Mixes are empty at the beginning of the simulation. The first message that is taken into account for the results is the one that arrives when the first input has been flushed with 99% probability. All messages flushed after the last arrival to the mix are also discarded for the results. This is done in order to eliminate the transitory initial and final phases.

5 Results

In this section we present and analyze the results we have obtained with the simulations.

5.1 Analysis of the input traffic

It is a common assumption in the literature that the arrivals to a mix node follow a Poisson process. We have analyzed the input traffic, and found that it does not follow a Poisson distribution, nor any other known distribution.

A Poisson process is modeled by a single parameter λ representing the expected amount of arrivals per (fixed) time interval. If the arrivals to a mix are assumed to follow a Poisson process with an average of λ arrivals per time interval Δt and we denote the number of arrivals in such a time interval by X , then X is Poisson distributed with parameter λ : $X \sim \text{Poiss}(\lambda)$. Important to note is that λ is *time-independent*.

In our statistical analysis we first *assumed* that the process of arrivals *is* a Poisson process and we estimated the parameter λ . The latter was done by taking the maximum likelihood estimate given the number of arrivals per time interval $\Delta t = 15$ minutes ($N = 11800$). We also constructed a 95% confidence interval for this estimate. Then we performed a goodness-of-fit test: can we reject the hypothesis

$$H_0 : \text{the number of arrivals per time interval} \sim \text{Poiss}(\bar{\lambda}) ,$$

where $\bar{\lambda}$ varies over the constructed confidence interval. The goodness-of-fit test we used is the well-known Chi-square test. Using a significance level of 0.01, the null hypothesis gets rejected!

In the right part of figure 2 we show the number of messages arrived to the mix per hour. The left part (below) of figure 2 shows the evolution of the arrivals per day. We can observe that the traffic arrived to the mix during the first month is much heavier than in the following three months. This shows that the input traffic pattern that gets to a mix node is highly unpredictable.

Figure 3 shows the frequency in hours and in days of receiving a certain number of arrivals. We can see that in most of the hours the mix receives less than 20 messages.

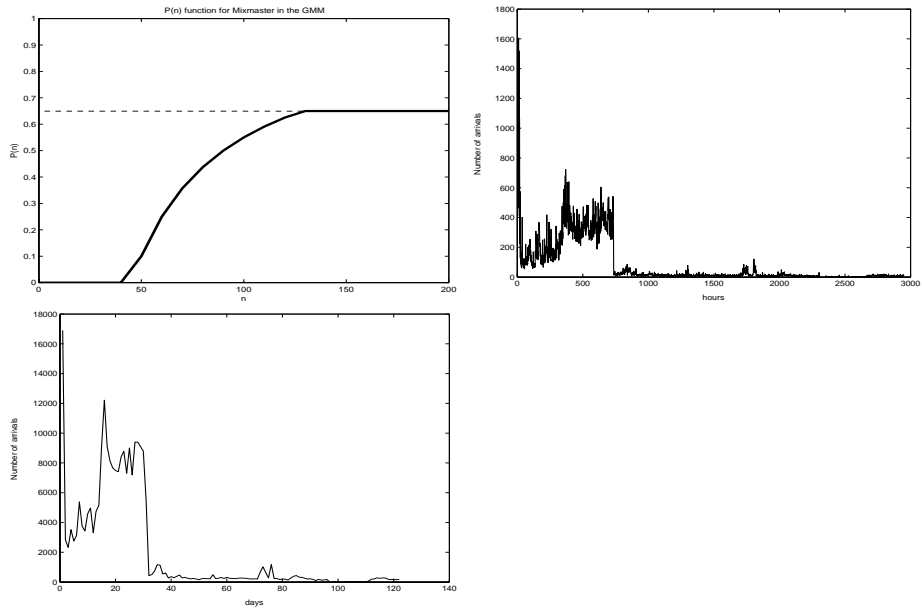


Figure 2: Incoming traffic patterns

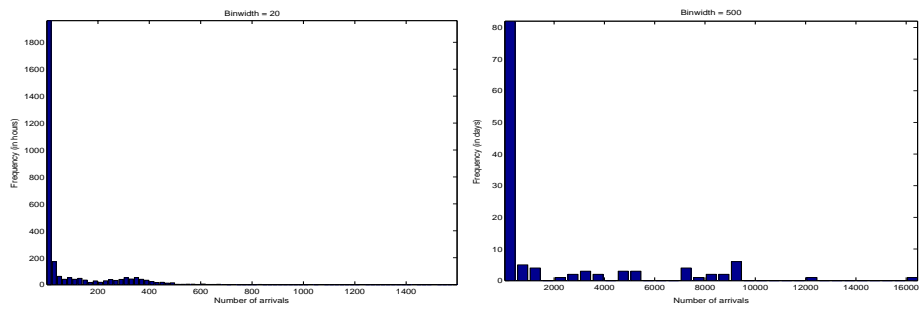


Figure 3: Frequency analysis of inputs

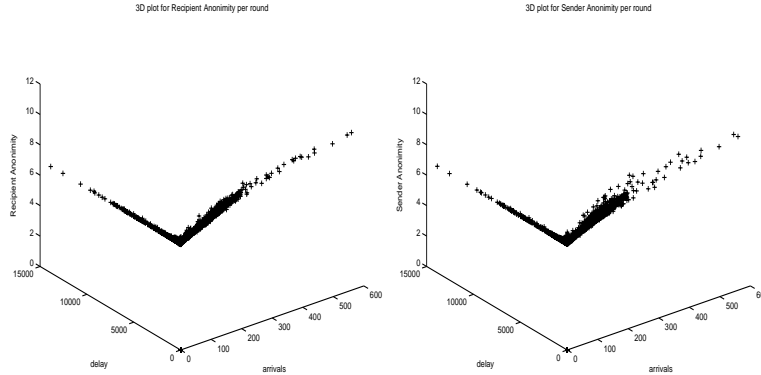


Figure 4: Recipient (left) and sender (right) anonymity for Mixmaster

5.2 Analysis of Mixmaster

We have simulated a Mixmaster node as explained in Section 4. Mixmaster is a pool mix and processes messages in batches. The recipient anonymity of all inputs of a round is the same. Equivalently, all outputs of a round have the same sender anonymity value. In this section we show the results obtained in our simulation.

In figure 4 we draw a point per round², located in the space defined by the delay (amount of time the message spends on the mix), arrivals (number of inputs received by the mix together with the message) and recipient anonymity of the message. Figure 4 shows the same for sender anonymity.

The first conclusion we come to when observing the figures is that there is a lower bound to the anonymity of Mixmaster. It is worth noting that, so far, we do not know any theoretical analysis of pool mixes able to predict the anonymity it provides, and prior to this analysis there were no figures on the anonymity that Mixmaster was actually providing. With this simulation, we can clearly see that Mixmaster guarantees a minimum sender and recipient anonymity of about 7. This means that the sender (recipient) of a message gets a minimum anonymity equivalent to perfect indistinguishability among $2^7 = 128$ senders (recipients).

We can see that the minimum anonymity is provided when the traffic (arrivals) is low. As the traffic increases, anonymity increases, getting maximum values of about 10 (i.e., equivalent to perfect indistinguishability among $2^{10} = 1024$ senders or recipients). We also observe that the delay of the messages doesn't take high values, unless the traffic load getting to the mix is very low.

In order to study the behaviour of the mix under different traffic loads, we have plotted values of delay and anonymity obtained in the simulation for the

²In pool mixes all messages of the same incoming round have the same recipient anonymity, and all the messages of the same outgoing round have the same sender anonymity

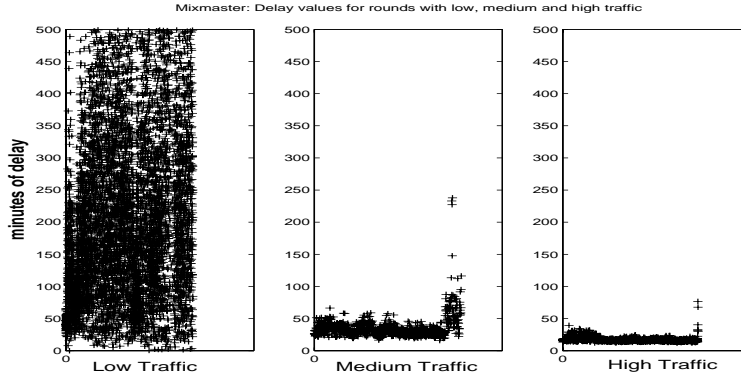


Figure 5: Delay values for Mixmaster

rounds with few arrivals (low traffic), intermediate number of arrivals (medium traffic) and many arrivals (high traffic).

We have selected the low, medium and high traffic taking into account the data statistics of the arrival process:

Low traffic: all rounds where the number of arrivals was between the first and third quantile; hence 50 percent of the rounds are denoted as normal traffic.

Medium traffic: all rounds where the number of arrivals was greater than the third quantile but lower than the outlier bound.

High traffic: all rounds with outlier values for the incoming messages.

In figure 5 we show the minutes of delay of every message. We can see that the delay only takes high values when the traffic is low. The fact that some messages appear as having a delay close to zero in the low traffic figure is due to the fact that we have more samples, so there are messages that arrive just before the flushing and are forwarded immediately. In figure 6 we show the recipient anonymity of every message (the sender anonymity presents very similar characteristics). We can see that as the traffic increases, the anonymity provided to the messages takes higher values.

5.3 Analysis of Reliable

The theoretical method proposed in [KEB98] that gives a probabilistic prediction on the anonymity provided by Reliable is based on the assumption of Poisson traffic. As we have seen, this assumption is definitely not correct for mix traffic.

We have simulated a Reliable mix as explained in Section 4. Reliable is a continuous (stop-and-go) mix. Reliable treats every message independently:

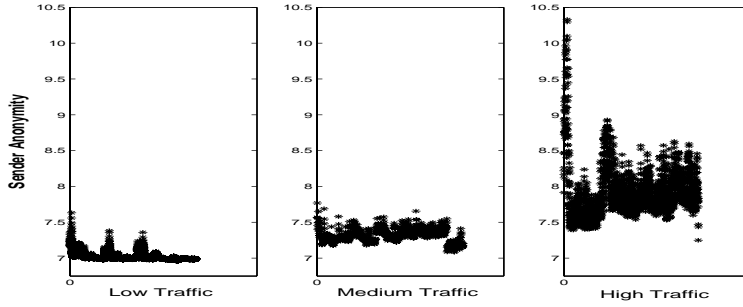


Figure 6: Anonymity values for Mixmaster

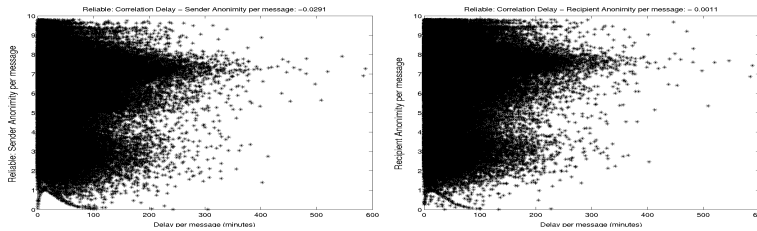


Figure 7: Correlation Delay-Anonymity for Reliable

when it gets a message it delays it a pre-determined amount of time (picked from an exponential distribution) and then forwards it. We represent a star, \star , per message.

In figure 7 we present the sender and recipient anonymity provided by Reliable for the real stream of inputs we have considered. We can see that the anonymity takes minimum values close to zero, which means that some of the messages can be trivially traced by a passive attacker. The maximum values of Reliable’s anonymity for this input are lower than Mixmaster’s maximums. Figure 8 shows the correlation of sender and recipient anonymity. These values are highly correlated (as in the case of Mixmaster). We can clearly see that some of the messages get nearly no anonymity.

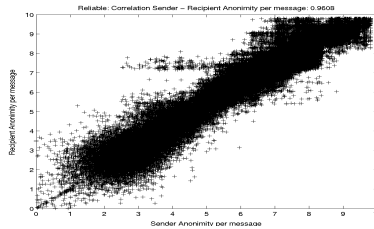


Figure 8: Correlation Sender-Recipient Anonymity for Reliable

5.4 Mixmaster vs. Reliable

As we have shown in the previous two sections, Mixmaster and Reliable have very different behaviours for the same traffic stream. Note that we have modified the default (1 hour) mean delay of Reliable in order to make a fair comparison between the two mixes (so the average delay of all messages is the same for the two mixes).

Mixmaster prioritizes the anonymity over the delay, and it provides a minimum recipient (sender) anonymity of around 7, equivalent to perfect indistinguishability among $2^7 = 128$ input (output) messages. When the traffic load decreases, Mixmaster provides a larger latency in order to keep the anonymity in high levels.

Reliable delays messages according to an exponential distribution, regardless of the traffic load. This has an effect in the anonymity, that will only have high values when there is a high traffic load. When the traffic load decreases, the anonymity provided by Reliable goes down to very low values. In some cases of very low load, Reliable does not provide anonymity at all.

Our conclusion is that a continuous mix like Reliable is not appropriate to provide anonymous services for applications that do not have real-time requirements (like email). A pool mix like Mixmaster should be used instead.

Continuous mixes like Reliable may be useful for real-time applications with tight delay constraints (like web browsing). Nevertheless, in order to provide acceptable levels of anonymity, the traffic load should be kept high.

6 Other factors which influence anonymity

We have evaluated the anonymity strength of the mixing algorithms implemented in Mixmaster and Reliable. Additional factors have a direct impact on the anonymity provided by the system. Concerns such as the security of the underlying operating system, host server integrity, proper implementation of the cryptographic functions provided by the remailer software, and likelihood of administration mistakes all contribute to the overall anonymity these software packages can provide. We assume that no active attacks against the software occurred during the development or compilation process, though additional concerns are present in that area [Tho84].

This paper does not aim to be an in-depth analysis of the full spectrum of host-attacks against remailer nodes, Nevertheless, it is important to mention some significant differences between Reliable and Mixmaster which may affect their ability to provide adequate anonymity for their users.

6.1 Host server integrity

The security of an operating mix is dependent on the security of the underlying host server. Many factors can impact the underlying system's security. Some considerations include shared access to the system by untrusted users, access

to key material on disk or in memory, and the ability to insert shims to attack dynamically loaded libraries called by the remailer software.

Reliable is limited to operation on the Windows platform. Mixmaster is portable, and has been known to run on a wide variety of operating systems.³

Host server security is ultimately the responsibility of the remailer operator.

6.2 UI Issues

In privacy application client, an intuitive user interface is essential in order to ensure that the software is used consistently and correctly [Sas02]. A greater level of skill can safely be assumed when designing privacy software which is intended to be operated as a service, however. Most anonymity systems, including mix-net implementations, do imply a significant degree of complexity. Due to the fact that the operation of a public Internet service involves the correct configuration and maintenance of the host server, this necessary complexity is acceptable as long as the operator's skill level is sufficient. The level of skill required to properly install, configure, and operate a mix-net node should not exceed that required to properly install, configure, and operate the server itself.

The software packages we evaluated differed with regard to their interface complexity in a number of areas.

In general, Reliable has a greater "ease of use" factor with respect to its interface. Mixmaster automates many important tasks, such as adaptive dummy generation, key rotation and key expiration announcement, and integrates more easily with the host MTA⁴. Reliable's installation process is easier, but its build process requires the use of third-party commercial applications and assumes experience with Windows development, so most users will install a pre-compiled binary. Compilation of Mixmaster is performed through a simple shell script.

At first glance, it appears that Reliable will be easier for hobbyists to operate than Mixmaster. However, Mixmaster's difficulty does not rise above the difficulty of maintaining a secure Internet-connected server, and thus has little effect on the overall security of a mix node deployment. Hobbyists unable to maintain Mixmaster are equally unlikely to be able to secure their host server.

6.3 Programming language

While the most critical factor in the creation of secure code is the manner in which it is written, some languages lend themselves to greater risk of exploitable mistakes. An inexperienced or unskilled programmer will always be able to make an application insecure. The choice of programming language merely sets the bar for the required level of experience and ability necessary to develop applications in that language safely. Thus, when evaluating the likelihood of the

³There have been instances of remailers based on the Mixmaster 3.0 codebase operating on SunOS, Solaris, SunOS, AIX, Irix, BeOS, MacOS X, Windows NT (natively and through the use of Cygwin), Windows 2000 (natively and through the use of Cygwin), Windows XP (through the use of Cygwin), FreeBSD, NetBSD, OpenBSD, and multiple versions of Linux.

⁴Mail Transport Agent, e.g. sendmail or postfix

existence of exploitable code in an application, it is worthwhile to consider the programming language used to create that application. Since neither Mixmaster nor Reliable were written by seasoned software developers, we assume a level of experience which would allow for simplistic security mistakes.⁵

6.4 Included libraries

In addition to the standard POSIX libraries provided by the compilation OS, Mixmaster 3.0, the version of Mixmaster evaluated in this paper, requires that the zlib [DG96] and OpenSSL [CEHL] libraries be included. Optionally, Mixmaster also links against pcre [Haz] and ncurses [BHRPD].

Reliable requires many native Windows system calls as well as the third-party application, Mixmaster 2.0.4.

6.5 Cryptographic functions

Both Mixmaster and Reliable avoid direct implementation of cryptographic algorithms when possible. Mixmaster 3.0 relies strictly on OpenSSL for these cryptographic functions. Any attackable flaws in the cryptographic library used to build Mixmaster which affect the security if the algorithms⁶ used by Mixmaster may be an attack against Mixmaster as well.

Reliable abstracts the cryptographic operations one step further. To support the Mixmaster message format, Reliable acts as a wrapper around the DOS version of Mixmaster 2.0.4. Thus, any attack against the Mixmaster message format due to implementation flaws in Mixmaster 2.0 will work against Reliable as well. Mixmaster 2.0.4 relies on the cryptographic library OpenSSL or its predecessor SSLeay for the MD5, EDE-3DES, and RSA routines.⁷

6.6 Entropy sources

The quality of the entropy source plays an extremely important role in both the pool mix and S-G mix schemes. In pool mix systems, the mixing in the pool must be cryptographically random in order to mix the traffic in a non-deterministic way. The timestamps used to determine how long a message should be held by an S-G mix implementation must also be from a strong entropy source for the

⁵The bulk of the code for Mixmaster 3.0 was written by Ulf Möller as his first major software development project while completing his undergraduate computer science degree [Mö2]. He has since gained respect as a skilled cryptographic software developer for his open source and proprietary development projects. Reliable was authored under a pseudonym, and we can only speculate about the level of experience of its author. (There has been no known communication with the author of Reliable since February, 2000).

⁶It is understood that flaws in the cryptographic algorithms will affect the security of software which relies upon those algorithms. However, since most attacks on cryptographic applications are due to flaws in the implementation, care must be taken when evaluating the shared cryptographic libraries.

⁷Prior to the expiration of the RSA patent, versions of Mixmaster 2.0 offered support for the RSAREF and BSAFE libraries as well. Use of these versions of Mixmaster is largely abandoned.

same reasons. In addition, the Mixmaster message format specifies the use of random data for its message and header padding.

All software is dependent on its underlying operating system for a good source of entropy. Cryptographic quality entropy is a scarce resource on most systems⁸, and therefore the entropy sources provided by most modern operating systems actually provide PRNG output which has been seeded with truly-random data.

Mixmaster uses OpenSSL’s `rand_` functions⁹. Reliable uses the standard Windows system call, `Rnd()`, when obtaining entropy, with the exception of message and header padding (which is done by the supporting Mixmaster 2.0.4 binary). The `Rnd()` function is not a cryptographically strong source of entropy. `Rnd()` starts with a seed value and generates numbers which fall within a finite range. Previous work has demonstrated that systems which use a known seed to a deterministic PRNG are trivially attackable [GW96, Cor]. While its use of `Rnd()` to determine the latency for a message injected into the mix is the most devastating, Reliable uses `Rnd()` for many other critical purposes as well.

6.7 Network timing attacks

Packet counting, deduction of pool variables by timing observation. Affects pool-mixes more than s-g mixes, possibly aids attacker in some non-host based active attacks such as $(n - 1)$ attacks. The anonymity strength of a remailer should not require pool values to be hidden, and countermeasures to this class of active attacks should be taken. [DS03a]

7 Conclusions and future work

In this paper we have analyzed the traffic pattern of a real traffic stream going through a working mix node and found that the traffic is not Poisson, as it is commonly assumed in the literature. The traffic pattern is highly unpredictable. Therefore, no assumptions on the traffic should be made when designing a mix.

We measure the anonymity of the pool mix scheme used in Mixmaster by applying a metric previously proposed in the literature. We provide our own metric for evaluating the anonymity of the S-G mix variant used in Reliable which does not assume a Poisson traffic pattern.

Our comparison of the two predominant mixing applications shows that Mixmaster provides superior anonymity, and is better suited for the anonymization of email messages than Reliable. Mixmaster provides a minimum level of anonymity at all times; Reliable does not. Reliable’s anonymity drops to nearly zero if the traffic is very low. In high-traffic situations, Mixmaster provides a higher maximum of anonymity than Reliable for the same stream of input: 10.5

⁸Systems which employ the use of noisy diodes, or other plentiful sources of entropy have less of a concern for entropy pool exhaustion.

⁹OpenSSL relies on its internal PRNG seeded with various system sources to provide cryptographically strong entropy.

of Mixmaster versus 10 of Reliable. We have shown that Mixmaster provides higher average anonymity than Reliable for the same input and same average delay. Due to its nature as a pool mix, Mixmaster provides higher delays than Reliable in low traffic conditions. Comparatively, due to the nature of S-G Mixes, Reliable’s delay is not dependent on the traffic.

In addition, we have identified a number of key points of attack and weakness in mix software to which anonymity software designers need to pay particular attention. In addition to the areas of theoretical weakness which we have identified, we discovered a fatal flaw in the use of randomness in Reliable, which diminishes its ability to provide anonymity, independent of our findings with regard to the S-G mix protocol.

We can conclude from our analysis of the mixing algorithms used by these mix implementations that S-G mixes are not suitable for use with systems that may have occurrences of low traffic on the network. While S-G mixes are an appropriate solution to low-latency applications such as web mixing, pool mixes should be used for higher latency systems with fluctuating traffic loads.

Acknowledgments

Claudia Díaz is funded by a research grant of the K.U.Leuven. This work was also partially supported by the IWT STWW project on Anonymity and Privacy in Electronic Services (APES), and by the Concerted Research Action (GOA) Mefisto-2000/06 of the Flemish Government.

Evelyne Dewitte is a research assistant with the I.W.T. (Flemish Institute for Scientific and Technological Research in Industry). Research supported by Research Council KUL: GOA-Mefisto 666, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects, G.0240.99 (multilinear algebra), G.0407.02 (support vector machines), G.0197.02 (power islands), G.0141.03 (identification and cryptography), G.0491.03 (control for intensive care glycemia), G.0120.03 (QIT), research communities (ICCoS, AN-MMM); AWI: Bil. Int. Collaboration Hungary/ Poland; IWT: PhD Grants, Soft4s (softsensors), Belgian Federal Government: DWTC (IUAP IV-02 (1996-2001) and IUAP V-22 (2002-2006)), PODO-II (CP/40: TMS and Sustainability); EU: CAGE; ERNSI; Eureka 2063-IMPACT; Eureka 2419-FliTE; Contract Research/agreements: Data4s, Electrabel, Elia, LMS, IPCOS, VIB.

The authors also want to thank Jasper Scholten for looking at the feasibility of some simulation algorithms, Peter Palfrader for assisting with the gathering of input data for our simulations, and members of The Shmoo Group for discussion of secure programming issues.

A Method to compute the anonymity of Reliable

To formalize the behaviour of the mixes, we define:

- X_s : an incoming message arriving at time s ;
- Y_t : an outgoing message leaving at time t ;
- D : the amount of time a message has been delayed.

We know that the mixes delay the messages exponentially and we have set the mean to 43 minutes: $D \sim \exp(1/43)$:

$$\begin{aligned} \text{pdf : } f(d) &= \frac{1}{43} e^{-\frac{1}{43}d} && \text{for all } d \geq 0 ; \\ &= 0 && \text{elsewhere ;} \\ \text{cdf : } F(d) &= P(D \leq d) = 1 - e^{-\frac{1}{43}d} && \text{for all } d \geq 0 ; \\ &= 0 && \text{elsewhere .} \end{aligned}$$

All delay times are independent.

Crucial to note in this setup is that the sequence of outgoing messages is not a Poisson process. This would only be true if all inputs would arrive at the same time, hence belong to the mix when the delaying starts. But in our case, messages arrive at distinct moments in time, each being delayed upon their arrival times.

Mixes flush at fixed time moments which are observed by the attacker:

$$t \in \{\text{out}_1, \text{out}_2, \dots, \text{out}_M\}.$$

He also observes the arrival times:

$$s \in \{\text{in}_1, \text{in}_2, \dots, \text{in}_N\}.$$

If a message leaves the mix at time t , what are then the probabilities for the arrival times? Suppose the departure time $t = \text{out}$ is fixed. We then look for the probability that the message that left at time out is the same message as the one that entered the mix at time s :

$$P(Y_{\text{out}} = X_s) = P(D = \text{out} - s) .$$

We can hence rephrase the problem in terms of the delay: which values for the delay times are the most probable? Clearly, negative delay is impossible so only arrival times prior to out are probable. These arrival times form a set $\{\text{in}_1, \text{in}_2, \dots, \text{in}_k\}$ with $\text{in}_k < \text{out}$. The matching delay times are then $\{\text{out}-\text{in}_1, \text{out}-\text{in}_2, \dots, \text{out}-\text{in}_k\}$ to which we will refer to as $\{d_1, d_2, \dots, d_k\}$. Note that $d_1 > d_2 > \dots > d_k$. We are almost at the solution as the density function of the delay times is known! Caution has to be taken however as the exponential function is a continuous function which means that the probability of the delay taking a single value is zero: $P(D = d_1) = \dots = P(D = d_k) = 0!$

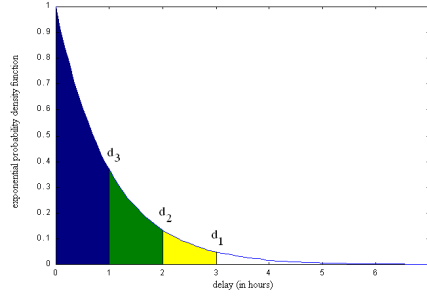


Figure 9: An example of an exponential probability density function

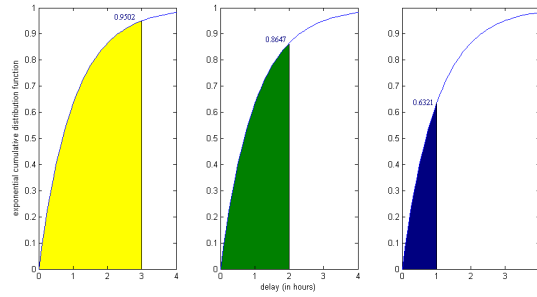


Figure 10: The matching exponential cumulative density function

How can we then calculate the probabilities of the delay times? To make this clear, let us look at figure 9 and suppose that we only have three arrival times prior to *out*. We have thus three possible delays $d_1 > d_2 > d_3$. Let us now assume for simplicity reasons that $d_1 = 3$ hours, $d_2 = 2$ hours and $d_3 = 1$ hour. The variable delay is continuous and can theoretically take every value in the interval $[0, 3]$. However, we know that we only flush at three particular times and that hence only three particular delays can occur. We can exploit this knowledge in the following way:

$$\begin{aligned}
 P(D = d_1) &\approx P(d_2 < D \leq d_1) = \text{yellow surface} ; \\
 P(D = d_2) &\approx P(d_3 < D \leq d_2) = \text{green surface} ; \\
 P(D = d_3) &\approx P(0 < D \leq d_3) = \text{blue surface} .
 \end{aligned}$$

In this way one can clearly see that the biggest surface corresponds to the most probable delay! This is straightforward for more than three delays. For computation we make use of the cumulative distribution function (cdf) which is graphed in figure 10. Cumulative probabilities are listed in tables and known in statistical software. For reasons of simplicity we put the mean of the exponential

to be 1 hour (easy parameterization):

$$\begin{aligned}P(D = d_1) &\approx F(d_1) - F(d_2) = 0.9502 - 0.8647 = 0.0855 ; \\P(D = d_2) &\approx F(d_2) - F(d_1) = 0.8647 - 0.6321 = 0.2326 ; \\P(D = d_3) &\approx F(d_3) = 0.6321 .\end{aligned}$$

In our little example, the message corresponds most likely with the one that entered the mix 1 hour before *out*. You can also clearly see this on figure 9. In practical applications however, many possible delays will occur so that visual inspections will not be efficient and calculations have to be made and compared.

A.1 Uniform Delays

Reliable allows for mix-chosen uniform delays if the users do not specify any delay for their messages.

We have found a method to compute the anonymity provided by a mix that delays inputs uniformly from a distribution $U[a, b]$. The method consists in creating a tables with all inputs and outputs. Then we search for all possible combinations input-output that are possible from an external observer's point of view (i.e., those that assign to every input that arrives at time T an output that leaves between $T + a$ and $T + b$). Let us call the total number of combinations C .

Then, in order to compute the recipient (sender) anonymity of message m_i , we need to find the distribution of probabilities that link this input (output) to all outputs (inputs).

If input m_i appears matching output s_j in P cases, then the probability assigned to s_j is P/C .

The probability of an input of matching an output is computed as possible cases divided by total cases. From this distribution, the sender and recipient anonymity can be computed for every message.

Unfortunately, due to the large amount of messages considered, the implementation of this algorithm in our case is not feasible.

References

- [BHRPD] Z. Ben-Halim, E. Raymond, J. Pfeifer, and T. Dickey. Ncurses.
- [CEHL] M. Cox, R. Engelschall, S. Henson, and B. Laurie. The OpenSSL Project.
- [Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2):84–88, February 1981.
- [Cor] Microsoft Corporation. Visual basic language reference–rnd function. *MSDN Library*.

- [Cot] Lance Cottrell. Mixmaster and remailer attacks.
- [Cot95] Lance Cottrell. Announcement: Mixmaster 2.0 remailer release! Usenet post, May 1995.
- [DDM03] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.
- [DG96] P. Deutsch and J-L. Gailly. ZLIB Compressed Data Format Specification version 3.3. Request for Comments: 1950, May 1996.
- [DP04] Claudia Diaz and Bart Preneel. Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In *Accepted submission at IH2004*, 2004.
- [DS03a] George Danezis and Len Sassaman. Heartbeat traffic to counter (n-1) attacks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003)*, Washington, DC, USA, October 2003.
- [DS03b] Claudia Diaz and Andrei Serjantov. Generalising mixes. In *Privacy Enhancing Technologies*, LNCS, Dresden, Germany, April 2003.
- [DSCP02] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.
- [GW96] Ian Goldberg and David Wagner. Randomness and the Netscape browser. *Dr. Dobb's Journal*, January 1996.
- [Haz] Philip Hazel. Perl compatible regular expressions.
- [KEB98] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.
- [Mö2] Ulf Möller. Personal communication. Private email to Len Sassaman, August 2002.
- [MCPS03] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2, July 2003.
- [PK00] Andreas Pfitzmann and Marit Kohntopp. Anonymity, unobservability and pseudonymity — a proposal for terminology. In *Designing Privacy Enhancing Technologies: Proceedings of the International Workshop on the Design Issues in Anonymity and Observability*, pages 1–9, July 2000.

- [RPr99] RProcess. Selective denial of service attacks. Usenet post, September 1999.
- [Sas02] Len Sassaman. The promise of privacy. Invited talk, LISA XVI, November 2002.
- [SD02] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.
- [Tho84] K. Thompson. Reflections on trusting trust. *Communications of the ACM*, 27(8), August 1984.