An Analysis of Parallel Mixing with Attacker-Controlled Inputs

Nikita Borisov

UC Berkeley

Abstract. Parallel mixing [7] is a technique for optimizing the latency of a synchronous re-encryption mix network. We analyze the anonymity of this technique when an adversary can learn the output positions of some of the inputs to the mix network. Using probabilistic modeling, we show that parallel mixing falls short of achieving optimal anonymity in this case. In particular, when the number of unknown inputs is small, there are significant anonymity losses in the expected case. This remains true even if all the mixes in the network are honest, and becomes worse as the number of mixes increases. We also consider repeatedly applying parallel mixing to the same set of inputs. We show that an attacker who knows some input–output relationships will learn new information with each mixing and can eventually link previously unknown inputs and outputs.

1 Introduction

Re-encryption mixes [6, 10, 9, 8] are a kind of mix network [1], where each mix server re-encrypts each input ciphertext, producing an equivalent encryption of the plaintext that is unlinkable to the original. Such mix networks avoid the requirement of key agreement with the mix servers prior to sending a message, as the re-encryption operation can happen without knowing the decryption key; they have applications in electronic elections, but they could also be used in place of regular mix networks. Synchronous mix networks require that each mix server permute the entire set of inputs in sequence; in contrast, asynchronous mix networks pass different inputs to different servers freely. Synchronous mix networks avoid some of the attacks on asynchronous networks [12], but do so at the cost of performance, as each server must re-encrypt the entire set of inputs and the others must wait and be idle while it does so.

Parallel mixing [7] is a technique to speed up synchronous re-encryption mix networks while attempting to preserve their anonymity guarantees. It divides the input into batches, with each server mixing the inputs in its own batch and then passing it to other servers. The scheme parallelizes the mixing workload among all servers, increasing the per-server computation cost but dramatically lowering the total mixing time. Parallel mixing can be made secure even if all but one of the mixes are compromised, matching the security of conventional synchronous mix networks.

However, the design of parallel mixing is such that not all possible permutations of the mixed inputs are generated; therefore, when some relationships between inputs and outputs are known to the attacker, parallel mixing leaks information about the other inputs. This may happen either because the attacker controls some of the inputs and can therefore track the outputs, or when the attacker learns which inputs correspond to outputs through other means. For example, attacks such as traffic analysis or intersection attacks can help the attacker identify some of the input–output correspondences. In this paper, we set out to investigate exactly how much information is revealed by parallel mixing in this case.

We use both the anonymity metric from [7], as well as the common entropy-based metric [11, 4] to quantify anonymity. We develop two approaches to measure anonymity of parallel mixing: a probabilistic simulation, computing exact distributions of the metric, and a sampling technique that approximates the distributions, which is useful for larger mix network sizes. We find that parallel mixing falls significantly short of achieving the same anonymity levels as conventional mixing in the expected case, and in some cases, such as with few unknown inputs and many parallel mixes, reveals a lot of information about the correspondence of inputs to outputs.

We further show how an attacker can use this information when the same set of inputs are mixed repeatedly using parallel mixing. (Such a situation might occur if parallel mixing is used to provide privacy for long-term communication.) Each instance of parallel mixing is essentially an independent observation, and the attacker can combine the information from all observations to accurately pinpoint which input corresponds to which output after a small number of rounds. This attack re-introduces the anonymity degradation properties of asynchronous mix networks [12] into parallel mixing, and is effective even when *none* of the mix servers are compromised.

The anonymity shortfall we describe may not apply to the electronic election application of parallel mixing. In particular, many elections can ensure that most inputs are not controlled by the attacker and that the same inputs are not mixed multiple times. However, the speed improvements of parallel mixing may make them attractive for other applications, such as anonymous email or web surfing, where our assumptions are valid and the problems we describe are practical. Our hope is to caution against the use of parallel mixing in such applications, unless one can ensure that the attacks we describe do not apply.

The following section provides some background on parallel mixing. Section 3 analyzes the anonymity of parallel mixing when some inputs are known to the attacker. Section 4 describes how this information can be used to discover which input corresponds to which output after several repeated rounds of mixing. Finally, Section 5 concludes and discusses some future research directions.

2 Background

2.1 Parallel Mixing

The parallel mixing technique described by Golle and Juels relies on breaking the inputs into batches and then successively passing the input batches between servers for re-encryption. We proceed to give an overview of their technique; please refer to [7] for more details. (For clarity, we will use the same terminology as Golle and Juels wherever possible.) Consider a network of M re-encryption mixes operating on n inputs. We will assign inputs to individual *slots*, and each mixing round will move the input ciphertexts between slots. For symmetry, we require that $M^2|n$. Parallel mixing is parameterized by a threshold $M' < M$, which is the maximum number of compromised mix servers.

The first step is to assign the input ciphertexts randomly to slots. The random permutation is defined from a public, ideal source of randomness (in practice, it would be computed jointly by all the servers). The slots are then partitioned into batches, $S(1), \ldots, S(M)$ of equal size, with each batch assigned to an individual mix server. Then the batches undergo $M' + 1$ mixing steps and M' rotation steps. In a mixing step, each mix permutes the inputs among the slots in the batch assigned to it. A rotation step involves passing batches between servers in succession, so server i passes its batch to server $i + 1 \pmod{M}$.

After this, a distribution step follows. In this step, the inputs in each batch are redistributed so that an equal number ends up in each resulting batch. I.e. for each original batch S_i and new batch S'_j, $|S_i \cap S'_j| = n/M^2$. After distributing the inputs in this way, there are another $M' + 1$ mixing steps, with M' rotation steps in between.

If we label the input batches as $B(1), \ldots, B(M)$ and the output batches as $C(1), \ldots, C(M)$, then the first step ensures that each ciphertext is assigned to a random batch $B(j)$. Then the batch $B(j)$ is mixed by $M' + 1$ servers, at least one of which must be honest. Therefore, before the distribution, the slot that an input i occupies within a batch j is chosen uniformly at random, and is unknown to the corrupt mixes. Then in the distribution step, i is assigned to an effectively random output batch $C(j')$. Finally, the next $M' + 1$ mixing steps ensure that the output batch $C(j')$ is once again mixed by at least one honest server, and hence the position of the input within the batch is unknown.

Following this process, each input ciphertext is equally likely to end up in each of the output slots. Golle and Juels show that if no more than M' servers are compromised, and no input–output relationships are known, the attackers cannot learn any information about the correspondence of the mix inputs and outputs.

Therefore, we discount mix corruption attacks and in fact we will assume that all the mixes are honest for the remainder of this paper. Instead, our focus will be on situations where the attacker learns some input–output relations, either through submitting rogue inputs to the mix or by other means. In this case, Golle and Juels suggest that the anonymity is statistically close to optimal. We will proceed to quantify the difference between parallel mixing and an optimal mix and examine the consequences of such a difference.

2.2 Anonymity

To perform a meaningful analysis, we need to have some measure of anonymity. Golle and Juels define an anonymity measure of a network as

$$Anon = \left(\min_{k,j} Pr(i_k \rightarrow o_j) \right)^{-1}$$

where i_k are input positions of the parallel mix and o_j are output positions. Since we are concerned with the anonymity achieved when an attacker knows some of the input output relations, the minimum should be taken over those inputs and outputs that the attacker does not know. The intuition for this measure is that when $Anon = n$, the worst-case probability of a true input–output relationship being guessed is $1/n$, or equivalent to a uniform mixing among n input–output pairs. Thus, with n unknown input–output relationships, we would like $Anon$ to be as close to n as possible.

In addition to this measure, we will use an entropy-based metric, proposed in [11, 4] and used to analyze many anonymous systems [2, 5, 3]. The metric involves computing a probability distribution X of inputs corresponding to a particular output (or vice versa), and computing the entropy of this distribution using the formula $H(X) = \sum_i -p_i \log_2 p_i$, where $p_i = Pr[t \rightarrow i]$, the probability that a target input t gets mapped to output slot i during mixing. (We will consider the problem of linking a given input to its corresponding output; the converse problem is analogous due to symmetry inherent in the mixing process.) The intuitive interpretation is that the metric represents the number of bits of uncertainty that an attacker has about the relationship between inputs and outputs. The entropy measure is also useful for certain kinds of information-theoretic analysis, which we will explore below.

The two anonymity measures are also connected by the relation:

$$H(X) \geq \log_2 Anon$$

3 Anonymity Analysis

We first motivate our analysis by a simple example. Consider a parallel mix network with $M = 3$ and $n = 9$. Initially, the 9 inputs are permuted and assigned into 3 batches of size 3. Then each batch is permuted, the inputs are redistributed into new batches, and these batches are permuted again before being output. Since we are assuming that all mixes are honest, we can assume that each batch will undergo a perfectly random permutation and therefore we can ignore the order of the inputs in each input batch, as well as the order of outputs in each output batch. Therefore, we can simplify the problem to considering which input batches the inputs get assigned to by the initial permutation (which an attacker can observe), and which output batches each input gets distributed to (which the attacker cannot see).

Suppose now that the attacker knows the input–output relations for all but 2 of the inputs. How much anonymity will parallel mixing provide the other 2? Consider the initial permutation; with probability 3/4, the two inputs will be assigned to different batches. Therefore, some batch B will contain the unknown input i_1 as well as two attacker-known inputs a_2, a_3. After the distribution step, the inputs in batch B will be distributed among the 3 output batches. The key point here is that each output batch will have exactly *one* input from B. Therefore, if the attacker can observe the position of the outputs a_2 and a_3, he can learn which output batch C contains i_1.

The other unknown input i_2, which we assumed was in some other input batch, will be assigned to some output batch by the distribution process. With probability 2/3, it will be a batch $C' \neq C$. In that case, the other two inputs in C will be attacker-known inputs a_4, a_5. This will allow the attacker to immediately identify i_1 as the member of C and therefore determine which output it corresponds to. Combining the two probabilities, we see that in $3/4 * 2/3 = 1/2$ the cases, the parallel mixing provides *no* anonymity to the two inputs. (In the other half of the cases, the attacker does not learn anything about I_1 and I_2.)

The foregoing is an extreme example, but it helps illustrate the kind of potential problems introduced by parallel mixing. In essence, although parallel mixing can assign any input slot to any other output slot, it generates only a subset of the permutations on all inputs. Therefore, knowing the relationship between some of the inputs and their permuted positions allows an attacker to deduce information about the other inputs. We now proceed to formally analyze the extent of such information for larger mix sizes and more unknown inputs.

3.1 Previous Results

Golle and Juels show how an attacker that knows some input–output relations can use this information to estimate probabilities of unknown inputs and outputs being linked through the mix. Consider $A(I)$ be the set of inputs known to the attacker, and $A(O)$ be the set of the corresponding outputs. Let $\alpha(j) = |B(j) \cap A(I)|$ be the number of slots in input batch j occupied by known inputs and $\gamma(j') = |C(j') \cap A(O)|$ be the number of slots in output batch j' occupied by known outputs. Also, let $\delta(j, j') = |B(j) \cap A(I) \cap C(j')|$ be the number of inputs in input batch j known to the attacker that are mapped to output batch j'. Then [7, Theorem 4.2] states:

Theorem 1 *Let $s_0 \in B(j)$ and $s_1 \in C(j')$ with $s_0 \notin A(I)$ and $s_1 \notin A(O)$. Then:*

$$Pr(s_0 \to s_1) = \frac{n/M^2 - \delta(j, j')}{(n/M - \alpha(j))(n/M - \gamma(j')}$$

Theorem 1 shows that $Pr(s_0 \to s_1)$ is only dependent on $\alpha(j)$, $\delta(j, j')$, and $\gamma(j')$. Golle and Juels approximate $\alpha(j)$,$\gamma(j')$ by Poisson random variables, with mean of $|A(I)|/M$ and standard deviation $\sqrt{|A(I)|/M}$, and $\delta(j, j')$ by a random variable with mean $|A(I)|/M^2$ and standard deviation of $\sqrt{|A(I)|/M^2}$. When each of the variables is equal to their mean, Theorem 1 shows that the anonymity is optimal: $Pr(s_i \to s_j) = \frac{1}{N-|A(I)|}$. However, variations in the values of $\alpha(j), \delta(j, j')$, and $\gamma(j)$ are going to cause the anonymity to be lower. For example, Figure 1 plots $Pr(s_0 \to s_1)^{-1}$ when $\alpha(j), \delta(j, j')$ and $\gamma(j')$ are each one standard deviation away from the mean, when 1000 inputs are distributed among 5 or 10 mixes. There is a significant distance from optimal anonymity shown in this graph, which becomes larger as the number of mixes increase.

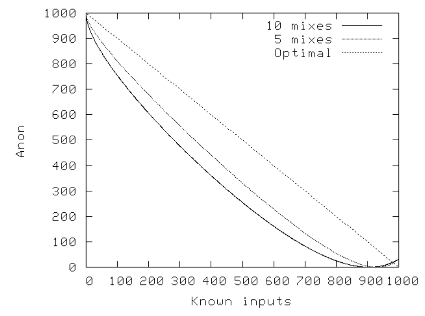

Fig. 1. Anonymity achieved at one standard deviation from the mean with 1000 inputs and 5 or 10 mix servers.

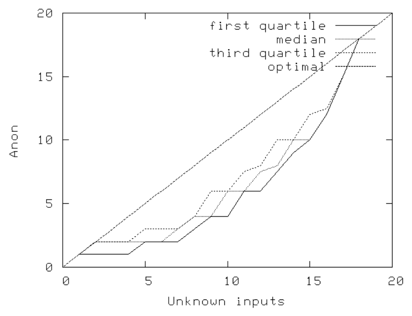

Fig. 2. Anonymity of 3 mixes with 18 inputs.

The Poisson model is only an approximation and does not accurately estimate how likely this scenario is, since the random variables are not in fact independent. In the rest of this section, we will use simulations to measure the possible values of $\alpha(j), \delta(j, j')$, and $\gamma(j')$ and the corresponding anonymity.

3.2 Simulation Results

We have built a probabilistic simulation of parallel mixing. At a high level, the simulation consists of taking inputs $i_1, \ldots, i_n$ and assigning them to input slots. Then we non-deterministically simulate each of the steps in parallel mixing; we compute each possible resulting assignment of inputs to slots and record the probability of arriving at each assignment. Let I denote an ordering of the inputs after the initial permutation, and O denote their ordering after all the steps of parallel mixing. Our simulation allows us to compute $Pr[I \to O]$ for all pairs I and O.

Starting with some subset of attacker-known inputs, and input and output orderings I and O we can compute the anonymity measure $Anon(I, O)$ as follows: first, we determine the positions of the attacker inputs in I and O and use that to compute $\alpha(j), \gamma(j, j')$, and $\delta(j')$ for all j, j'. Then we apply Theorem 1 to compute $Pr[s_0 \to s_1]$ for all s_0, s_1 and take $Anon(I, O) = \min_{s_0, s_1} Pr[s_0 \to s_1]^{-1}$.

Using the results from our probabilistic simulation, we can then compute the expected anonymity by $\sum_{I,O} Pr[I \rightarrow O]Anon(I, O)$. We can also use them to compute the median or other measures on the distribution of anonymity.

Simulating all the permutations of inputs becomes impractical very quickly. Fortunately, for the purposes of computing the *Anon* metric, we can make a few simplifying assumptions. First, since we are assuming at most M' mixes are corrupt, we can model the mixing and rotation steps by a uniformly random permutation of each individual batch. Second, an initial distribution with a different ordering of batches, or a different order of inputs within a batch, produces an identical distribution of outputs. Similarly, the order of inputs in the output batch does not affect the variables α, γ, δ, hence we can stop the simulation after the distribution step. Finally, we can treat all unknown inputs as identical, and all known inputs within a given mix as identical, greatly reducing the space of possible permutations.

With these simplifications, we are able to model mix networks of moderate sizes. Figure 2 shows the median as well as the first and third quartile values for the *Anon* metric calculated on a mix network with 3 mixes and 18 inputs. Even with only one known input, the *Anon* metric falls short of optimal, and in almost all the cases, the median value of the metric is significantly below the maximum. For example, with 9 unknown inputs, the median value for *Anon* is 4 meaning that in over half the cases, there exist s_0 and s_1 such that $Pr[s_0 \rightarrow s_1] \geq \frac{1}{4}$, instead of the $\frac{1}{9}$ we would hope for with an optimal system.

3.3 Sampling Based Results

The probabilistic simulation methodology does not to scale to mix networks of large sizes. In this case, we use sampling to get an estimate of what kind of anonymity to expect in such networks. Instead of simulating all possible permutations of inputs, we instead compute the results of a mix network using random permutations and apply the anonymity metric to that. We repeat this multiple times to obtain sampling of the distribution of the *Anon* metric. The estimate is inexact and will not capture the tail of the distribution (even with the smaller network sizes, we observed events of probability $< 1\%$). However, it is representative of what users of the mix network should expect to see in practice.

Figure 3 shows the cumulative distribution function of the *Anon* function on a mix network with 1008 inputs, 900 of which are unknown. The CDF was estimated using 1000 samples; the figure demonstrates the effect of different numbers of mix servers on the anonymity of the network. Unsurprisingly, the anonymity degrades with a larger number of servers. With more mix servers, the permutations that are generated are more restricted, as the distribution step forces the inputs to be directed to one of a larger number of batches. However, what is perhaps surprising is the amount of anonymity loss. With 12 servers, the median value of *Anon* is nearly one fifth lower than optimal.

The difference is even more dramatic when the attacker knows more input–output relationships. Figure 4 shows the CDF corresponding to a network with

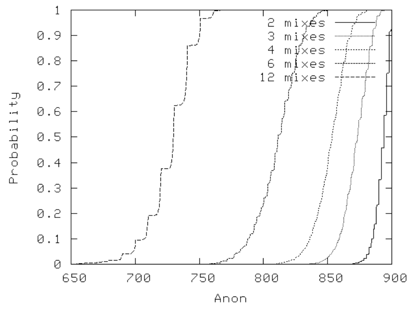

Fig. 3. Sampled anonymity CDF with 1008 inputs, 900 unknown, and 2–12 servers.

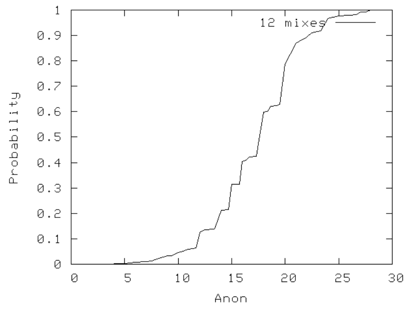

Fig. 4. Sampled anonymity CDF with 1008 inputs, 100 unknown, and 12 servers.

12 servers and 1008 inputs, 100 of which are unknown. The median value for *Anon* is only 18, and the largest we observed after 1000 trials was only 30. This figure shows that introducing parallelism into a mix system where one can expect an attacker to know a large fraction of the input–output relationships greatly reduces the anonymity provided by this system.

3.4 Entropy Metric

We can use the same techniques to compute the entropy-based anonymity metric. Both probabilistic simulations and sampling let us calculate $Pr[s_0 \rightarrow s_1]$ for each s_0, s_1. Therefore, given an input t, we can compute the probability $Pr[t \rightarrow s_1]$ for each slot s_1 and take the entropy of the resulting distribution.

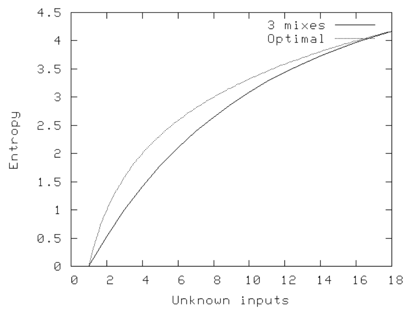

Fig. 5. Expected entropy with 3 mixes, 18 inputs.

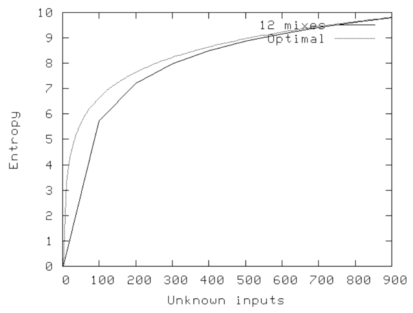

Fig. 6. Sampled expected entropy with 12 mixes, 1008 inputs.

Figure 5 shows the expected entropy for a 3 mix network with 18 inputs. Note that the entropy will be larger than $\log_2 Anon$ for two reasons: the entropy metric takes into account the entire probability distribution, rather than the highest value, and the expectation is taken over a particular input, rather than the worst-case input for a given mix, as is the case with *Anon*. Therefore, the metric is more "forgiving" than *Anon*; however, there is still a significant difference between the optimal anonymity and what is achieved by parallel mixing, especially when most inputs are known to the attacker.

Figure 6 shows the expected entropy obtained by sampling a 12 mix network with 1008 inputs. Once again, the difference from optimal entropy is more significant when more of the inputs are known to the attacker. These results suggest that parallel mixing should not be used in situations where an attacker might be expected to learn many input–output correspondences, either through controlling the inputs themselves or through some outside source of information.

4 Multi-Round Anonymity

In the previous section, we showed how parallel mixing fails to achieve optimal anonymity. However, in most cases, the attacker gains only a statistical advantage over the optimum, but is unable to directly link an input with an output. In this section, we show how the attacker can use this statistical advantage over repeated rounds of mixing to reveal previously unknown correspondence between inputs and outputs. Such repeated mixing may occur when parallel mixing is used to protect long-term communication, such as a regular email correspondence or a set of long-lived TCP connections.

4.1 Repeated Mixings

To begin, consider the input to the parallel mix consisting of a set of unknown inputs, $G(I)$ and a set of attacker-known inputs $A(I)$. And let's imagine the attacker wants to determine which output $i_0 \in G(I)$ corresponds to. The attacker can observe the initial permutation to find out which slot i_0 is assigned to, as well where the other inputs in $G(I)$ and $A(I)$ are assigned, and then observe the positions of $A(I)$ in the output. Let us call this entire observation $\mathcal{O}_1$. Now consider a particular output o_k; let $s_0 \in B(j)$ be the slot assigned to I_0 and $s_1 \in C(j')$ be the slot of o_k. The attacker can compute $\alpha(j), \delta(j,j')$, and $\gamma(j')$ and then derive $Pr(s_0 \to s_1)$ using Theorem 1. We can write that $Pr[i_0 \to o_k|\mathcal{O}_1] = Pr(s_0 \to s_1)$. The attacker can compute this value for each k efficiently, since all that's necessary is $\alpha(j), \delta(j,j')$, and $\gamma(j')$ for each pair j, j'.

Now suppose that the same set of inputs is sent to the mix a second time, with a different initial permutation and a different mixing process. Consider the observations he makes in this round represented by $\mathcal{O}_2$. Then, once again, the attacker can compute $Pr[i_0 \to o_k|\mathcal{O}_2]$. $\mathcal{O}_1$ and $\mathcal{O}_2$ are *independent* observations, meaning that:

$$Pr[\mathcal{O}_1 \wedge \mathcal{O}_2|i_0 \to o_k] = Pr[\mathcal{O}_1|i_0 \to o_k] \cdot Pr[\mathcal{O}_2|i_0 \to o_k]$$

Using this fact, we can show that:

$$Pr[i_0 \to o_k|\mathcal{O}_1 \wedge \mathcal{O}_2] = \frac{Pr[i_0 \to o_k|\mathcal{O}_1] \cdot Pr[i_0 \to o_k|\mathcal{O}_2]}{\sum_{j=1}^{|G(I)|} Pr[i_0 \to o_j|\mathcal{O}_1] Pr[i_0 \to o_j|\mathcal{O}_2]} \tag{1}$$

(See Theorem 2 in Appendix A for details.) Now, in the optimal anonymity case, $Pr[i_0 \to o_j|\mathcal{O}_l] = \frac{1}{|G(I)|}$ for each $l = 1, 2$, $j = 1, \ldots, n$. In this case, $Pr[i_0 \to o_j|\mathcal{O}_1 \wedge \mathcal{O}_2] = \frac{1}{|G(I)|}$, i.e. the attacker learns no new information from repeated mixes. However, as we saw in the last section, we expect the anonymity to fall somewhat short of optimal with parallel mixing, in which case the attacker will be able to amplify the anonymity loss with repeated observations.

For example, consider the case where $|G(I)| = 2$, $Pr[i_0 \to o_1|\mathcal{O}_l] = 0.6$, and $Pr[i_0 \to o_2|\mathcal{O}_l] = 0.4$, for $l = 1, 2$. Then $Pr[i_0 \to o_1|\mathcal{O}_1 \wedge \mathcal{O}_2] \approx 0.69$. Therefore, given 2 observations, the attacker has more confidence that $i_0 \to o_1$ than from each individual observation.

We can extend (1) to a set of observations $\mathcal{O}_1, \ldots, \mathcal{O}_n$:

$$Pr\left[i_0 \to o_k \,\middle|\, \bigwedge_{l=1}^{n} \mathcal{O}_l \right] = \frac{\prod_{l=1}^{n} Pr[i_0 \to o_k|\mathcal{O}_l]}{\sum_{j=1}^{|G(I)|} \prod_{l=1}^{n} Pr[i_0 \to o_j|\mathcal{O}_l]} \tag{2}$$

If there is, as we showed in the last section, a bias in the probability distributions based on each observation towards the true o_k, this bias will be amplified with multiple observations and eventually reveal the true correspondence with a high confidence.

4.2 Simulations

We can measure the success of this attack by using simulations. The simulation set up is similar to that of Section 3.3. We simulate a mixing of a set of inputs and record an observation $\mathcal{O}_1$. Then, for a particular input i_0, we compute the probability distribution $Pr[i_0 \to o_k|\mathcal{O}_1]$ for each k. Then we perform another trial to obtain another probability distribution. After a number of trials, we apply (2) to compute the probability $Pr[i_0 \to o_k| \bigwedge O_l]$ for each k.

Figure 7 shows the success of this attack for a mix network with 12 mixes and 1008 inputs total, with varying numbers of them being unknown. We plot the probability P_{right} that is assigned to the true output corresponding to i_0, and P_{wrong}, which is the highest probability assigned to each incorrect guess. Initially, there is insufficient information to identify the correct correspondence; however, after a sufficient number of rounds, P_{right} invariably tends to 1 and P_{wrong} to 0. For 100 unknown inputs, fewer than 10 rounds are required to identify the correct output. The attack remains effective even with larger numbers of unknown inputs: with 500 unknown inputs, fewer than 100 rounds successfully identify the correct link. However, as the number of unknown inputs increases, the success of the attack diminishes; with 900 unknown inputs, the most likely guess for the link is incorrect even after 700 rounds of mixing.

We can use information theory to predict how quickly this attack succeeds. Section 3.4 shows how to compute the expected entropy metric applied to parallel

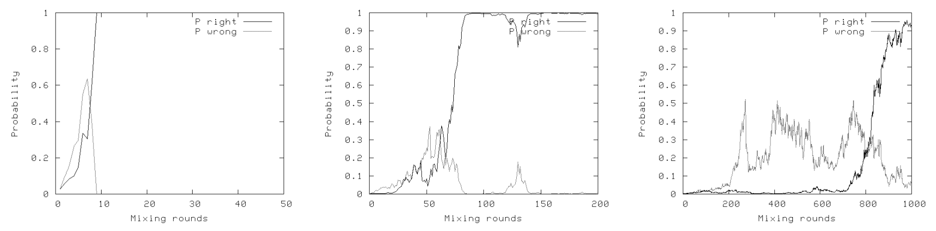

Fig. 7. Success of repeated mixing attack with 100, 500, and 900 unknown inputs.

mixing. The expected entropy is also known as *conditional entropy*, or $H(X|O)$, where X is a random variable modeling input–output correspondences, and O is modeling observations. The conditional entropy is closely related to the mutual information between X and O:

$$I(X;O) = H(X) - H(X|O)$$

If we consider parallel mixing as a noisy communication channel, $I(X;O)$ shows how many bits of information are revealed with each mixing. Therefore, to identify a particular output among $n - |A(I)|$ unknown ones, we will need at least $\log_2(n - |A(I)|)/I(X;O)$ rounds of mixing.

With $n - |A(I)|$ unknown inputs, $H(X) = \log_2(n - |A(I)|)$. We can therefore calculate the number of rounds required to reveal the input–output correspondence based on the data in Figure 6. For 100 unknown inputs, we expect to need about 7 rounds of mixing. With 500 unknown inputs, the expected number of rounds is 91, and with 900, it is as high as 880. Therefore, we can see that with 900 unknown inputs, the attack is unlikely to succeed unless very many rounds of mixing are performed, while with smaller numbers of unknown inputs, the attack can be quite effective.

Information theory offers only a lower bound on the number of rounds needed, and potentially more rounds will be required. However, the data in Figure 7 show that the lower bound comes close to being tight.

5 Conclusions and Future Work

We have presented an analysis of parallel mixing and how well it performs when the attacker can learn the relationship between some inputs and outputs. We showed that in such cases, there is a significant difference between the anonymity provided by parallel mixing and the optimal anonymity achieved by conventional mixes. In addition, we demonstrated how this difference may be exploited to reveal the secret mapping of inputs to outputs when the same inputs are mixed repeatedly by way of parallel mixing.

Note that our attacks apply even when *all* the mix servers are honest. Further, they do not require that the attackers control any of the inputs, but rather only that they know some of the input–output correspondences. Such information may be revealed through other attacks, such as traffic analysis or intersection attacks. Hence a completely passive adversary may be able to compromise the security of parallel mixing. Therefore, we strongly caution against using parallel mixing in situations when attackers may learn some input–output correspondences and/or when the same inputs are mixed multiple times.

For a more complete understanding of parallel mixing, it would be useful to analyze directly the success of combining intersection attacks or traffic analysis with our techniques for exploiting the information revealed by parallel mixing. Such an analysis would show whether such attacks are practical in a given setting, and thus whether parallel mixing is appropriate. Unfortunately, our current simulation techniques face a state explosion problem preventing such analysis.

An important question is whether parallel mixing can be extended to correct the problems we present while maintaining some of the performance advantage. For example, it can be shown that adding another distribution and a rotation/mixing step to parallel mixing will cause it to generate all permutations of the inputs, albeit with a non-uniform distribution. Once again, our current analysis methods can only analyze this extension for very small network sizes and cannot predict its resilience against our attacks.

Our analysis in Section 4.2 touches upon a connection between conditional entropy, mutual information, and the performance of multi-round attacks. Our information-theoretic model may apply to other anonymity systems; perhaps the conditional entropy metric can be used to derive useful bounds on the success of multi-round attacks, such as a generalization of the predecessor attack [12].

Acknowledgments

We would like to thank Marco Barreno, Rob Johnson, Umesh Shankar, and Naveen Sastry for initial discussions aboutu parallel mixing that motivated this paper, David Molnar for his thoughts on efficient enumeration, and the anonymous reviewers for their insightful comments.

References

1. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.
2. George Danezis. Mix-networks with restricted routes. In Roger Dingledine, editor, *Proceedings of the Privacy Enhancing Technologies (PET) Workshop*. Springer-Verlag, LNCS 2760, March 2003.
3. Claudia Diaz, Len Sassaman, and Evelyne Dewitte. Comparison between two practical mix designs. In *9th European Symposium on Research in Computer Security*, September 2004.
4. Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies (PET) Workshop*. Springer-Verlag, LNCS 2482, April 2002.

5. Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In *Proceedings of the Privacy Enhancing Technologies (PET) Workshop*, Toronto, Canada, May 2004.
6. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *CRYPTO*, pages 368–387, 2001.
7. Philip Golle and Ari Juels. Parallel mixing. In *ACM Conference on Communications and Computer Security*, October 2004.
8. Philippe Golle and Dan Boneh. Almost entirely correct mixing with applications to voting. In *ACM Conference on Communications and Computer Security*, pages 68–77, 2002.
9. Markus Jakobsson, Ari Juels, and Ron Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, pages 339–353, 2002.
10. C. Andrew Neff. A verifiable secret shuffle and its applications to e-voting. In *ACM Conference on Communications and Computer Security*, pages 116–125, 2001.
11. Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies (PET) Workshop*, San Diego, CA, April 2002. Springer-Verlag, LNCS 2482.
12. Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed Security Symposium (NDSS)*. IEEE, February 2002.

A Conditional Probabilities

Theorem 2 *Given some set of inputs $I = \{1, \ldots, n\}$ and some set of observations $\mathcal{O}$, let $\mathcal{O}_1, \mathcal{O}_2 \in \mathcal{O}$ be two* independent *observations on some input i_0. Then:*

$$Pr[i_0 = k | \mathcal{O}_1 \wedge \mathcal{O}_2] = \frac{Pr[i_0 = k|\mathcal{O}_1] Pr[i_0 = k|\mathcal{O}_2]}{\sum_{j=1}^{n} Pr[i_0 = j|\mathcal{O}_1] Pr[i_0 = j|\mathcal{O}_2]}$$

Proof. The independence assumption can be formalized as:

$$Pr[\mathcal{O}_1 \wedge \mathcal{O}_2 | i_0 = k] = Pr[\mathcal{O}_1 | i_0 = k] Pr[\mathcal{O}_2 | i_0 = k]$$

Then on one hand:

$$Pr[\mathcal{O}_1 \wedge \mathcal{O}_2 | i_0 = k] = \frac{Pr[\mathcal{O}_1 \wedge \mathcal{O}_2 \wedge i_0 = k]}{Pr[i_0 = k]}$$

On the other hand,

$$\begin{aligned} Pr[\mathcal{O}_1 \wedge \mathcal{O}_2 | i_0 = k] &= Pr[\mathcal{O}_1 | i_0 = k] Pr[\mathcal{O}_2 | i_0 = k] \\ &= \frac{Pr[\mathcal{O}_1 \wedge i_0 = k]}{Pr[i_0 = k]} \frac{Pr[\mathcal{O}_2 \wedge i_0 = k]}{Pr[i_0 = k]} \end{aligned}$$

Therefore,

$$\frac{Pr[\mathcal{O}_1 \wedge \mathcal{O}_2 \wedge i_0 = k]}{Pr[i_0 = k]} = \frac{Pr[\mathcal{O}_1 \wedge i_0 = k]Pr[\mathcal{O}_2 \wedge i_0 = k]}{Pr[i_0 = k]^2}$$

$$Pr[\mathcal{O}_1 \wedge \mathcal{O}_2 \wedge i_0 = k] = \frac{Pr[\mathcal{O}_1 \wedge i_0 = k]Pr[\mathcal{O}_2 \wedge i_0 = k]}{Pr[i_0 = k]}$$

And:

$$Pr[\mathcal{O}_1 \wedge \mathcal{O}_2] = \sum_{j=1}^{n} Pr[i_0 = j \wedge O_1 \wedge O_2] = \sum_{j=1}^{n} \frac{Pr[\mathcal{O}_1 \wedge i_0 = j]Pr[\mathcal{O}_2 \wedge i_0 = j]}{Pr[i_0 = j]}$$

$$\begin{aligned}
Pr[i_0 = k|\mathcal{O}_1 \wedge \mathcal{O}_2] &= \\
&= \frac{Pr[i_0 = k \wedge \mathcal{O}_1 \wedge \mathcal{O}_2]}{Pr[\mathcal{O}_1 \wedge \mathcal{O}_2]} = \frac{\frac{Pr[\mathcal{O}_1 \wedge i_0 = k]Pr[\mathcal{O}_2 \wedge i_0 = k]}{Pr[i_0 = k]}}{\sum_{j=1}^{n} \frac{Pr[\mathcal{O}_1 \wedge i_0 = j]Pr[\mathcal{O}_2 \wedge i_0 = j]}{Pr[i_0 = j]}} \\
&= \frac{Pr[\mathcal{O}_1 \wedge i_0 = k]Pr[\mathcal{O}_2 \wedge i_0 = k]}{\sum_{j=1}^{n} Pr[\mathcal{O}_1 \wedge i_0 = j]Pr[\mathcal{O}_2 \wedge i_0 = j]} \\
&\quad \text{(because } Pr[i_0 = k] = Pr[i_0 = j] = \frac{1}{n} \text{ for all } j.) \\
&= \frac{\frac{Pr[\mathcal{O}_1 \wedge i_0 = k]}{Pr[\mathcal{O}_1]} \frac{Pr[\mathcal{O}_2 \wedge i_0 = k]}{Pr[\mathcal{O}_2]}}{\sum_{j=1}^{n} \frac{Pr[\mathcal{O}_1 \wedge i_0 = j]}{Pr[\mathcal{O}_1]} \frac{Pr[\mathcal{O}_2 \wedge i_0 = j]}{Pr[\mathcal{O}_2]}} \\
&= \frac{Pr[i_0 = k|\mathcal{O}_1]Pr[i_0 = k|\mathcal{O}_2]}{\sum_{j=1}^{n} Pr[i_0 = j|\mathcal{O}_1]Pr[i_0 = j|\mathcal{O}_2]}
\end{aligned}$$