[[1]](#footnote-1)

Effectiveness of Blending Attacks on Mixes

Meng Tang

*Abstract*—This paper presents an analysis of the effectiveness of blending attacks against mixes. We first introduce a generalized model for expressing mixes and blending attacks, which describes a mix as an (exponential) function of time and a blending attack as a series of mix flushes. Then we analyze the blending attack’s performance using this generalized model. We also analyze blending attacks against a mix that generates dummy messages..

*Index Terms*—Anonymity, Blending Attack, Mix

# INTRODUCTION

M

ixes are widely used in modern anonymity systems. Mixes provide anonymity by exploiting the indistinguishability between encrypted messages to break the link between incoming and outgoing messages [4]. The amount of anonymity that a mix can provide is measured by its anonymity set size. Mixes are subject to a variety of threats including replay, blending, pseudospoofing, tagging, intersection and timing [7]. This paper focuses on blending attacks, which is a general class of attacks that reduce anonymity set size by manipulating traffic flow.

The attacker of a blending attack is an active attacker who is able to monitor all incoming/outgoing messages, and delay/destroy/modify/replay/fabricate messages. Given sufficient power, an attacker is able to shape the network traffic in such a way that an output batch of the mix contains only the target message and spurious messages generated by the attacker. Then the attacker is able to identify the target message by filtering out spurious messages. This kind of attack is known as the “n-1” attack. The authors of [9] believe there is “no general applicable method to prevent this attack”.

The effectiveness of a blending attack is usually measured in terms of its resource cost and the anonymity set size it is able to achieve. Against a mix without memory, the attacker is able to reduce its anonymity set size to 1 with finite resource consumption.

The threat of blending attacks can be reduced by introducing memory, known as pools, into mixes. A pool allows a message to stay in the memory of the mix when the mix fires, with a possibility determined by parameters, thus making it more difficult for the attacker to flush out all legitimate messages. . This technique, however, also brings about an increased delay in message forwarding as a cost do defend against blending attacks [3].

Analysis of blending attacks against mixes with memory has been done in previous researches. [7] presents a precise analysis on the number of rounds required to perform a blending attack on a variety of pool mixes, and gives the possibility that the attack succeeds in a number of rounds. [3] shows a generalized framework for expressing the batching strategies of mixes, and analyzes the cost and the chance of success to attack a binomial mix.

This paper will present a more generalized description of mixes and blending attacks. Measurement of the cost to attack will be in terms of time instead of rounds, and chance of success will be measured in terms of anonymity set size.

# attack model

A blending attack involves a mix and a global active attacker.

The mix accepts encrypted input messages from various senders or other mixes, decrypts them, and output them in batches. Normally only the mix can establish an equivalency between incoming and outgoing messages. Note that if the attacker is trying to perform a per-message attack, then the input source and output destination are not relevant, so we can ignore the senders and receivers altogether and assume that the mix is connected to an input channel and an output channel.

Multiple kinds of mixes are available, including timed mixes, threshold mixes, timed pool mixes, threshold pool mixes, binomial mixes, etc. Usually mixes are considered to operate in “rounds”, however, we prefer using time instead, because time is more sensible than rounds when considered as a resource cost. We can also observe that timed mixes and threshold mixes are not essentially different, because the attacker can set a threshold mix’s firing rate by fixing the incoming traffic to a certain value. So we will treat threshold mixes and threshold pool mixes as timed mixes and timed threshold mixes.

In this paper, we will describe a mix with a single function . The definition of is as follows:

If a message is in mix at time , where belongs to , a subset of real numbers, then is the possibility that m remains in at time . We shall see later that this description applies for timed mix, threshold mix, timed pool mix, threshold pool mix and binomial mix.

The attacker selects one from the input messages as a target, and tries to find its counterpart in the output. In an (n-1) attack, it is typically achieved in 2 flushing steps. The attacker first flushes all legitimate messages from the mix, then let the target message enter the mix, and finally flushes the target message from the mix.

Typically the attacker is considered able to delay and insert messages. In this paper we will think differently. Since the attacker is able to indefinitely delay any incoming messages, and the attacker will always do so because new incoming messages will add more uncertainty to the attack, so we will not consider any legitimate incoming messages except the target message. Moreover, we are assuming that all mixes operate on a “time” basis, so the attacker will wait for the desired output instead of forcing a flush.

The attack model can be summed up as follows:

1. The attacker attacks at time 0, when there are a total of legitimate messages in the mix. Let be the last time that the mix fires before time 0. For each of these message , the possibility that m still resides in the mix at time is . is determined by the mix’s settings and the attacker’s computing power. There will be no incoming legitimate messages since .

2. The attacker waits until time , and puts the target message, , into the mix.

3. The attacker waits until , when at least one message is outputted from the mix. The attacker picks one from the output messages, and claims it is the target message.

Some assumptions are to be made toward this attack model:

1. The mix does not base its batching strategy on input sources.

2. The mix does not base its batching strategy on the time each message has stayed in the mix.

3. The attacker is interested in the linkage between input and output messages, not the sender’s identity.

# correctness of attack model

In this section, we will demonstrate the correctness of the attack model by fitting common mixes in this model.

Threshold mix and threshold pool mix:

I show that a threshold mix under blending attack can be transformed by the attacker into a timed mix. This fact is almost trivial: for a mix with threshold , the attacker sets its firing rate to by sending in a message in each time. The range of depends on and the attacker’s computing power. The same can be done to a threshold pool mix to convert it into a timed pool mix.

Timed mix:

Using traditional model, the steps of attacking a timed mix are as follows:

1. The attacker blocks all incoming messages at time 0, and waits until the mix fires at .

2. The attacker lets the target message go into the mix at .

3. The attacker waits until the mix fires at , and recognizes the only unknown message as the target

We can fit this attack strategy into the framework with these definitions:

Let be the last time that the mix fires before the attacker blocks all incoming messages.

Since and , all messages will have left the mix at .

Since is in the mix at , and , , so is not in the mix at . The attacker should be able to find the decrypted from the output between and , because it is the only unknown output message within the time period.

It is easy to verify that both models give the same time consumption - , and anonymity set size - 1.

Timed pool mix:

The attacking scheme is similar to that against timed mixes, except that the attacker may flush the mix for several consecutive rounds instead of just one.

The steps are as follows:

1. The attacker blocks all incoming messages at time 0, and keeps flooding the mix for consecutive rounds, until the mix fires for the th time at .

2. The attacker lets the target message go into the mix at .

3. The attacker keeps flooding the mix, until the mix outputs at least one unknown message at , and claims one of them to be the target.

In this case, we will again let be the last time that the mix fires before the attacker blocks all incoming messages, and give the following definitions:

: pool size

: number of spurious messages attacker inserts each round

: number of legitimate messages left in mix at

Anonymity set size is .

If there is a restriction on the total rounds that the attack may spend on step 3, then there is a chance that the attacker cannot complete the attack. If the limit is rounds, then

Binomial mix:

A binomial operates in a way similar to that of a timed pool mix, except that the “pool size” is dynamic. In each round, each message has a chance of to be flushed out. The attacker attacks a binomial mix in these steps:

1. The attacker blocks all incoming messages at time 0, and waits for consecutive rounds, until the mix fires for the th time at .

2. The attacker lets the target message go into the mix at .

3. The attacker waits until the mix outputs at least one unknown message at , and claims one of them to be the target.

We use the following definitions to fit it into our attack model:

: number of legitimate messages left in mix at

Anonymity set size is .

If there is a restriction on the total rounds that the attack may spend on step 3, then there is a chance that the attacker cannot complete the attack. If the limit is rounds, then

We further show that timed pool mixes and binomial mixes are special cases of our model that is an exponential function.

In the above analysis of timed pool mixes and binomial mixes, we have assumed that

But when considering a blending attack, most points of this function does not carry any meaning. An attacker smart enough will never end any step of the attack between mix fires. He either halts at the previous mix fire to save time, or move on to the next for better results. Therefore, we may redefine in the following form:

Then we construct an expansion of

So we consider them special cases where is an exponential function, and the attacker stops only at mix fires.

With some calculation, it is not difficult to apply our framework to other mix types such as threshold or timed mix, and threshold and timed mix. Other mix types may also arise from our framework. However this framework works only when the mix is under attack.

# analysis

Our analysis of blending attacks on a mix will be based on the following definitions and assumptions:

1. The mix does not operate in “rounds”, it may output any message at any time.

2. The mix treats all messages equivalently.

3. If a message is in the mix at time , then the chance that it is still in the mix at time is , or to say, .

4. The attacker blocks all incoming messages at , inserts the target message at , and ends the attack at . If at least one message unknown to the attacker is flushed between and , the attack is complete with the attacker claiming one of them to be the target, otherwise the attacker fails to complete the attack. We assume that there’re a total of legitimate messages in the mix at .

The performance of the mix will be measured by message delay, .

The effectiveness of the attack will be measured by anonymity set size, , and chance of not completing the attack, . The attacker’s chance of success is .

First we obtain an estimation of . Though the exact value of is unknown, we may get an expectation by assuming the mix is in “steady state” [7]. We assume that the mix has operated in a network with fixed incoming traffic and has reached steady state. Let be the input rate of messages, then , .

The anonymity set size is dependent solely on the number of legitimate messages left in the mix at . .

The attacker’s chance of success is

 has an upper bound of , which is decided by the first half of attack. In practice, the first half is more critical and time consuming than the second because the attacker needs to reduce the anonymity set size as much as possible in this step. Once the target message enters the mix, the attacker will have no means to further reduce anonymity set size. The second half only requires flushing out one non-spurious message. Whether this message is the target is not important, because the attacker cannot distinguish between them.

An attack is exact if . An attack is exact and certain if and . An exact certain attack requires that there exists some such that . Since is monotonically decreasing, it is possible only if there exists a value such that for any , we have , and it also indicates that is bounded by . An attack can be exact but uncertain if no such exists, but , and so the attacker can reduce to 1 after spending infinite resources. Note that should always hold, or there will be a positive possibility that a message remains in the mix indefinitely, therefore if no further mechanisms are introduced, an exact attack is always possible.

We show an example where is an exponential function,

By altering the parameter , we are able to reduce the attacker’s chance of success at a cost of increased message delay.

To demonstrate the relation between and , we plot as a function of (Fig. 1), with these parameter settings:

In this plot, we are treating expectations as exact values.



Fig. 1. Attacker’s chance of success as a function of the parameter l

The attacker’s chance to succeed increases with , and if breaks some vague “threshold”, the chance is almost 100%. It is better to keep as small as possible, but a smaller also means a greater message delay, which can be a headache.

Fig. 2 shows the relation between attacker’s chance to succeed and message delay, with the same parameters, and ranging from 0.1 to 5.



Fig. 2. Attacker’s chance of success as a function of message delay. Note that the unit of message delay is the same as the unit of t

We see that the attacker’s chance of success dramatically decreases when the message delay reaches a “threshold”. In this example, it seems reasonable to set the message delay to 2, so that the attacker has less than 10% percent chance to succeed. However, this setting is not actually favorable since the delay is comparable to the attacker’s time consumption.

Though not yet fully proved, our experiments show a trend that the threshold increases linearly as the attacker puts in more time. It is not unexpected since this property is easy to verify with timed mixes and timed pool mixes. This trend means that for each cost the mix spends to defend, the attacker just needs to put in some comparable effort to break the defense, which gives no advantage to the mix.

# mixes with dummy pool

Dummy messages have been used in other anonymity systems to provide ambiguity. The idea to use dummy messages to defeat blending attacks was first introduced by Danezis and Sassaman [11].

A dummy pool is a pool composed of dummy messages. Each time the mix is about to fire, messages in the dummy pool as well as normal messages are taken into consideration. The dummy pool is maintained by the mix itself, and is refilled whenever a dummy message is flushed, thus provides a reliable source of non-spurious messages. The dummy pool can be virtual, if dummy messages are generated on the fly.

Proposed mix:

We use the same parameters that we used to do former analysis, except that we are adding a dummy pool with pool size . As long as the function is not changed, the dummy pool has no effect on normal traffic, and will not increase message delay, so we still have the following:

Anonymity set size changes due to the dummy pool:

With the same set of parameters as above, the relation between message delay and the attacker’s chance of success is given in Fig. 3. Since the dummy pool has set 11 as a lower bound for the anonymity set size, the attacker has no more than 9.1% chance to succeed.



Fig. 3. Attacker’s chance of success as a function of message delay when dummy pool is introduced

This result almost too obvious. What we are most concerned is the cost to use a dummy pool. Let be the extra outgoing traffic incurred by dummy messages, then we have

 is the number of messages in a mix when the mix is in steady state, or pool size for a pool mix. If is large enough, can be set to decrease without imposing too much impact on the outgoing traffic.

The mix can also use a virtual dummy pool instead of actually maintaining one. With this setting, the mix generates and sends out dummy messages to random targets. For a virtual dummy pool of size , the output rate of dummy messages should be set such that the average dummy traffic is .

# conclusions

We present a model for describing blending attacks. In our model, mixes are defined by a single function of time. We have shown that when under a blending attack, timed mixes, threshold mixes, timed pool mixes, threshold pool mixes and binomial mixes can fit into the model. Our model ignores irrelevant factors such as spurious messages and pool size, and focuses on the number of legitimate messages left in the mix.

We also present an analysis on blending attacks based on this model. We evaluate the attack’s effectiveness by estimating the attacker’s chance to complete a successful blending attack. The results show that the outcome of the attack is largely dependent on the first step when the attacker tries to flush the pool. It is also shown that traditional mixes are always vulnerable to exact attacks. The relation between attacker’s chance of success and the mix’s message delay is plotted, and gives the conclusion that increasing message delay is not an efficient way to counter blending attacks.

Finally we repeat the calculations for mixes that utilize dummy messages. We describe the mix as a normal mix with an extra dummy pool, and again plot the attacker’s chance to succeed as a function of message delay. The result indicates that for a mix large enough, a dummy pool can defeat blending attacks at only a minor cost.

The main restriction of our model is that is applies almost only when the mix is under a blending attack, otherwise, some our simplifying assumptions may not hold. However, this model always works for binomial mixes.

References

1. Andreas Pfitzmann, Michael Waidner, “Networks Without User Observability”, in *Computers and Security*, 1987, pp. 158-166
2. Brian N. Levine, Michael K. Reiter, Chenxi Wang, Matthew Wright, “Timing Attacks in Low-Latency Mix Systems”, in *Financial Cryptography*, 2004, pp. 251-265
3. Claudia Diaz, Andrei Serjantov, “Generalising Mixes”, in *Privacy Enhacing Technologies*
4. David Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms”, in *Communications of the ACM*, 1981
5. David M. Goldschlag, Michael G. Reed, Paul F. Syverson, “Hiding Routing Information”, in *Proceedings of the First International Workshop on Information Hiding*, 1996, pp. 137-150
6. George Danezis, “Designing and attacking anonymous communication systems”, (UCAM-CL-TR-594)
7. Luke O'Connor, “On Blending Attacks For Mixes with Memory Extended Version”, in *7th International Workshop*, 2005, pp. 39-52
8. Oliver Berthold, Andreas P¯tzmann, Ronny Standtke, “The disadvantages of free MIX routes and how to overcome them”, in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pp. 30-45
9. Parvathinathan Venkitasubramaniam, Venkat Anantharam, “On the Anonymity of Chaum Mixes”, in *Proceedings 2008 IEEE international symposium on information theory*, 2008, pp. 534-538
10. Claudia Diaz, Bart Preneel, “Reasoning about the Anonymity Provided by Pool Mixes that Generate Dummy Traffic”, in *Information Hiding: 6th International Workshop*, 2004, pp. 309-325
11. George Danezis, Len Sassaman, “Heartbeat Traffic to Counter (n-1) attacks”, in *Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, 2003, pp. 89-93
1. This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, “This work was supported in part by the U.S. Depart­ment of Com­merce under Grant BS123456”.

The next few paragraphs should contain the authors’ current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp). [↑](#footnote-ref-1)