Enhancing chaum mixes with randomness

By Kannabiran Maheswaran

Introduction:

With the advancements of the internet, the need to hide oneself for personal reasons are very high. The knowledge of communication between two parties, regardless of the content of communication is very valuable. The internet has created a lot of safe methods of communication where one cannot tell with certainty that two parties have been communicating among themselves. One such method of communication would be the use of Chaum Mixes **[1]**

A mix is a middleman node that collects a bunch of messages from different parties and then after a certain amount of time fires all such collected messages. The point being it makes it a lot harder for one to trace a message from A to B seeing that all messages that go into the said middleman get jumbled up and sent out. However, they are not fool-proof. Attacks such as blending attacks reduce the anonymity of a mix to near zero. This paper looks at the anonymity provided by the design of a chaum mix that uses the generation of random variables to run.

The idea here is to introduce random factors into the mix making it unpredictable for an observer or an attacker to break.

Design:

The Mix's design plays a very big factor in increasing the anonymity provided by the mix and its resistance to blending attacks. The mix is going to consist of a bunch of functions that change priority upon every '*round'* and the random attributes that are assigned to said functions change every time the mix fires its messages.The mix has a random probability of calling each function Lastly the mix flushes all the messages that it has stored to signal a change of round.

Threshold Function:

A basic mix function where the mix fires upon recieving "**M**" messages. The mix will not fire before crossing the threshold. The anonymity provided by this function is directly related to the number of Messages that is assigned to the threshold based off the randomized attribute. Probability of a message being the message you want to check is :

$$\frac{1}{M+D}$$

where **D** is the number of dummy messages that have been added.

Timed Function:

The mix fires after a time period **t**. The number of messages fired '**T'** is equivalent to the number of messages received during the time period **t.**The anonymity provided by this function is again, directly related to the number of messages that are fired when the time period elapses. Probability of a message being the message you want to check is :

$$\frac{1}{T+D}$$

It is possible to receive zero messages during the time period causing the mix to fire just dummy messages.

Time and Threshold function:

The mix fires after a time period **t** and upon receiving **M** messages. Both these conditions must come to pass in order for the mix to fire. Additional conditionals can be added such that the mix fires if it crosses the time period twice over without receiving **M** messages.
The anonymity provided by this function is:

$$\frac{1}{F+D}$$

where **F** is the number of messages that were fired by the mix. **F** is never less than **M** unless additional conditions to fire before threshold have been imposed upon the function.

Pool Function:

The mix collects messages until it hits a certain threshold **P+Q** after which it fires **P** messages retaining the remaining **Q** messages in the 'pool'. If the function is called again within the same round, it selects a new value for **P** and **Q** and uses the messages that were left by the previous pool calling as part of the new calling.
The anonymity provided by this function is:

$$\frac{P}{P+Q} * \frac{1}{P+D}$$

This is the probability of the message not being left in the pool times the anonymity of a message from the messages fired.

Flush Function:

At the end of a round the mix fires all message that have accumulated in the system due to the calling of a Pool function. This ensures that a message does not remain in the system for too long providing a guarantee that a message will reach its destination after a certain time period unlike traditional Pool mixes.
The anonymity provided by this function is:

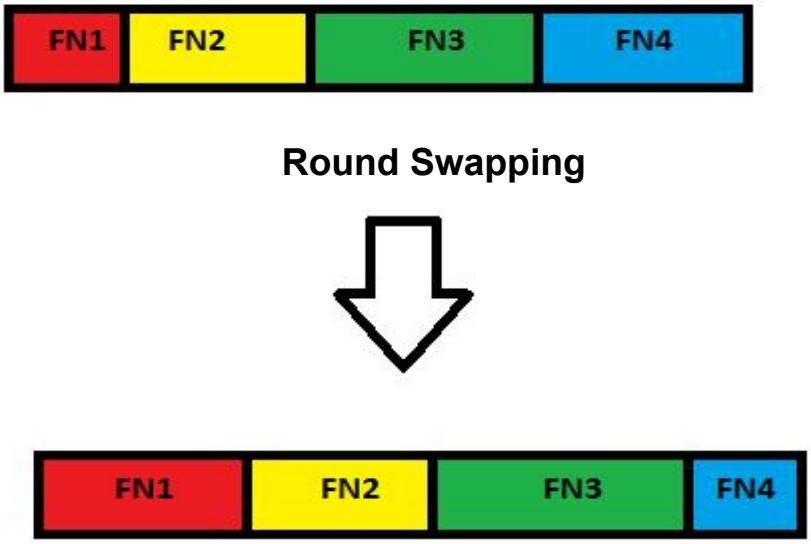
$$\frac{1}{F+D}$$

Where **F** is the number of messages that were left in the pool.

Round Functionality:

After the mix fires for a randomized amount of time, the mix will flush every message from its system and reorganise itself providing a different probability for each function that is to be called. This ensures that an adversary collecting information cannot determine what function

the mix would call with high accuracy. A simple cost effective way to change the probabilities of the functions would be to generate four random numbers. Each number by the sum of the four generated numbers would be the probability of that function being called.



Total Anonymity provided :

The total Anonymity provided by this system is tied to the probability of the pool function being called. In terms of an observer's perspective it would be one over the number of messages fired. But in actuality it is based off the pool function being called and the probability that the message was left in the pool and not fired.

$$1 - p(x) * \frac{1}{F+D} + p(x) * \frac{p}{P+Q} * \frac{1}{P+D}$$

where ***p(x)*** is the probability of a pool function being called. **F** is the number of messages fired along with the dummy messages.

Dummy messages:

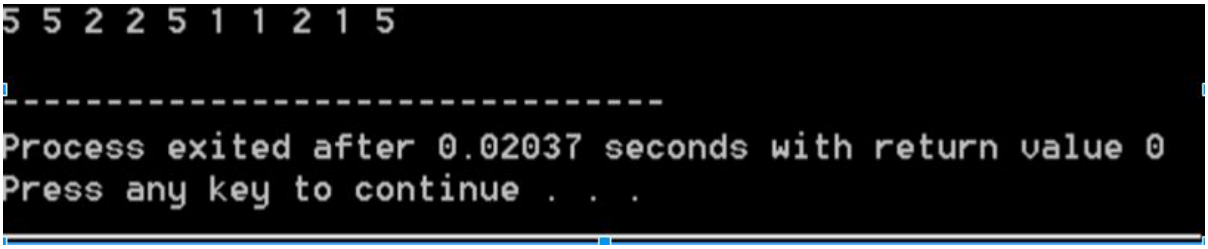
Dummy messages play a very important role in mixes. They are very useful in reducing end to end intersection attacks **[2]**. The first question here is, how much do we use and how expensive does it get? The average size of a message (let us go with email) is roughly 25 KB without an attachment and 500KB with an attachment **[3]**. Also, 75% of the messages are sent without attachments. Having a pool of dummy messages in the memory would reduce the time spent generating dummy messages if the mix is firing every few seconds. Depending on the parameters we would roughly need to keep about 10 times the number of dummy messages sent per fire. The mix could replenish the dummy pool while waiting to fire messages. So, for say an average of 20 dummy messages per fire, we would require to have a pool of 200 messages.

We can further increase efficiency by splitting the dummy Pool into larger dummy messages and smaller dummy messages. This allows the mix to quickly pad up dummy messages and other messages to the largest message that was received in that firing round.

The Dummy messages should be introduced to the mix at the output as it increases anonymity of a message and protects it from N-1 attacks. **[4]**

Cost of random Generation:

Considering that our design relies very heavily upon the generation of random numbers, we need to choose an algorithm that is quick,does not require a lot of entropy and at the same time not something that someone could predict with the collection of enough data.
*The Mersenne Twister***[5]** is the perfect pseudo-random number generator that fits our requirements. It is not cryptographically secure in its native form being that 624 iterations allows all future predictions, and so we use a modified Mersenne Twister known as *oriented Fast Mersenne Twister (SFMT)* which is twice as fast and cryptographically secure.**[6]**



The above figure is a test run for SFMT generator
0.02 seconds to generate a random number makes the cost of generation trivial.

Anonymity compared to a Pool mix:

The Randomized mix provides us with an anonymity of

$$1 - p(x) * \frac{1}{F+D} + p(x) * \frac{P}{P+Q} * \frac{1}{P+D}$$

per message that goes into the mix.
Meanwhile a standard Pool mix will provide us

$$\frac{P}{P+Q} * \frac{1}{P+D}$$

It can be noted that the difference is not very large considering that the first equation is a summation and probability of the function being called as compared to the second equation being just the function call for every message.

Attacks:

An attacker can never be exact with his attacks on the mix as the attributes of the mix function are constantly changing along with the very functions themselves changing. It is very hard to determine what function the mix is using. To an attacker a threshold mix could very well be a timed mix that fired at the time the attacker determined that an arbit value is the threshold, never mind the fact that the arbit value is different for the next fire.
An attacker can never be certain with his attacks because it cannot determine if a pool function was called, leaving the message in the system. The addition of dummy messages every time a mix fires ensures that the anonymity of a message can never reach zero even during an N-1 attack.
However, collection of data over a long period of time can give an observer the extremities of the random number that is generated giving them a slight semblance as to what the mix may

do.

Results:

It can be seen that the anonymity provided by this mix while not substantially more than using just a pool mix with dummy messages, provides us with uncertainty and inexactness at a trivial cost of random generation. It also provides us a guarantee that a message will leave the mix within a set amount of time due to the flushing function.

References:

[1] D. Chaum, “Untraceable electronic mail, return addresses and digital pseudonyms,” Communications of the ACM, vol. 24, no. 2, pp. 84–88, February 1981

[2] Berthold, Oliver and Langos, Heinrich (2002). Dummy traffic against long term intersection attacks. In Dingledine, Roger and Syverson, Paul, editors, Proceedings of Privacy Enhancing Technologies workshop (PET 2002). Springer-Verlag, LNCS 2482.

[3] Email Statistics Report, 2009-2013
Editor: Sara Radicati, Ph.D; Principal Analyst: Masha Khmartseva
The Radicati group INC

[4]Diaz, Claudia and Preneel, Bart (2004). Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In Accepted submission at IH2004

[5]Matsumoto, M.; Nishimura, T. (1998). "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator". ACM Transactions on Modeling and Computer Simulation 8 (1): 3–30. doi:10.1145/272991.272995

[6]SIMD-oriented Fast Mersenne Twister (SFMT): twice faster than Mersenne Twister
http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html