Mitchell Artin
Anonymity Fall 15
Final Project Report - Mix-Net with Loose Routing
December 15, 2015

Background

Mix-nets are a type of overlay network used to prevent traffic analysis over a standard network. They operate by passing encrypted network traffic over one or several “mixes” in between the sender and the recipient. The sender picks the route through mixes. On a mix-net, a local passive adversary (an actor that can only view traffic on a network link) cannot view the original source and the destination of a message, only the previous hop and the next hop. Each member of the mix-net has a public and private key, used to encrypt and decrypt data being sent to it.

Mix-nets can vary based the flushing algorithm used by the mixes. This determines how they send messages once they are received. One of these flushing algorithms defines a threshold and sends messages only when *n* messages are collected in a buffer. The resulting mix is called a threshold mix.

A common attack on mix-nets is called the *n-1* attack. An global active adversary (GAA, an actor that can view, delay, stop, and inject traffic on all network links) can use this attack to determine the overall route of a message. Threshold mixes are particularly vulnerable to this type of attack. To perform this attack, a GAA will single out a target message it wants to analyze. It will delay this message from reaching the next mix in its route. It will then wait to see when that mix flushes. Once the attacker knows the mix buffer is empty, it will allow only the target message to enter the mix, delaying all other messages. The GAA will then generate and inject *n-1* dummy messages, of which it knows the destination, into the mix. The mix will flush and the GAA will analyze where the traffic is sent. All destinations of dummy messages are known so the one message sent to an unknown destination will be the target message. This process can be repeated along the entire message’s path to reveal the original sender and destination.

Crowds are another type of overlay network used to confound a local passive adversary's traffic analysis by passing traffic along one or several “Jondos”. They differ from mix-nets as the sender does not pick a route for his or her data. Routes are chosen dynamically by the Jondos and each Jondo sees the destination in plain-text. The message’s next hop is chosen by the Jondo by generating a pseudo random number and comparing it with a set probability. The Jondo uses this number to decide whether to send the message to its final destination or another Jondo in the network. Symmetric key encryption is used between the Jondos.

Purpose

The purpose of this project is to take a standard mix-net and modify it with the goal of defeating the n-1 attack. The modifications aim to increase anonymity while providing minimal downsides to the user. Modifications will be inspired by the design of crowd networks.

Network Design

Like mix-nets, the sender chooses the route through the mixes that its message will take. Instead of following this fixed route, the mixes will implement a version of loose routing based on crowds. For each message received, the mix will generate a pseudo random number and compare it with a number set by the mix operator. If the generated number is less than the set number, the mix will chose a new next hop and add it to the route of the message. This number set by the mix operator can be considered the probability of choosing a new next hop. It is important to note that this new next hop is an addition to the route of the message. It is not a replacement for any of the mixes the sender originally chose to route the message with. Because of this, the new network design can guarantee that a message chosen to be routed with *x* mixes by the user will be routed with at least *x* mixes.

Use of Cryptography

Between each mix, asymmetric cryptography will be used. Public keys for all members of the network will be distributed to all members. Before a sender sends a message it will be encrypted for each hop along the way. First it will be encrypted with the private key of the destination. Then the address of the destination will be added to the end of that message and the resulting data will be encrypted with the private key of the last mix in the route. This process will continue, wrapping up the message with layers of encryption just like an onion, until it is encrypted using every private key of the chosen mix(es). This method allows each mix to see the next chosen hop/destination without seeing the complete route of the message or the message itself. The only member that has access to the message in plain-text is the recipient after the final decryption phase.

As the message travels along its route, it will be unencrypted with each mix's private key. After unencryption, the mix will see further encrypted data along with a next hop/destination in plain-text. This plain-text segment will be read and then stripped off. The remaining encrypted data will be sent to the next hop/destination.

If a mix chooses to add a new next hop for a message, it will generate a new pseudo random number to select a new next mix from a list of all available mixes in the network. It will then use that mix's public key to encrypt the message including the original next hop/destination. Note that in this case, the plain-text part of the unencrypted message is not stripped off as the new next hop will need this information.

Example

Environment

In this example, there are four mixes. The sender is named Alice and the recipient is named Bob. The test message to be sent is “Hey Bob, it's Mitch”. The probability that a mix will chose a new next hop is 50%. For simplicity the flushing algorithm is set to immediately send a message once a mix receives it, the threshold is set to n=1. The route through the mixes Alice has chosen is best represented by the following diagram.

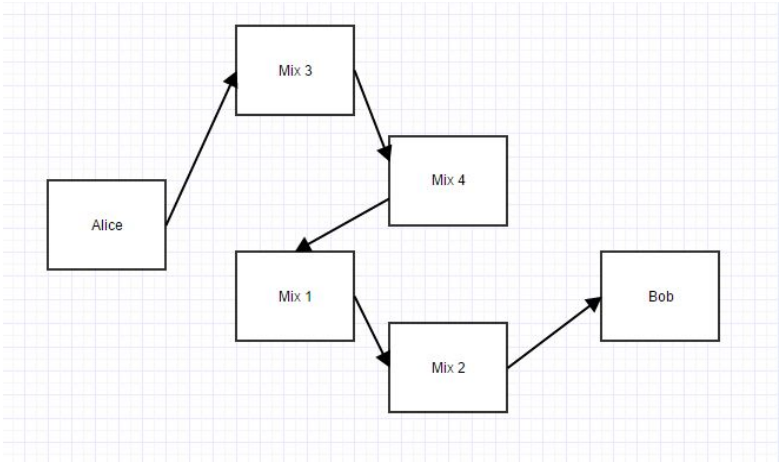

Figure 1. Planned route of message sent from Alice

This message and route is then inputted into the program and it is run. The events took place as follows.

- Alice sends message.
- Mix 3 receives message, generates a 0.67, sends to original next hop, Mix 4
- Mix 4 receives message, generates a 0.43, sends to new next hop, Mix 2
- Mix 2 receives message, generates a 0.23, sends to new next hop, Mix 1
- Mix 1 receives message, generates a 0.11, sends to new next hop, Mix 3
- Mix 3 receives message, generates a 0.81, sends to original next hop from before this “detour” started (Mix 4 -> Mix 1), Mix 1
- Mix 1 receives message, generates a 0.92, sends to original next hop, Mix 2
- Mix 2 receives message, generates a 0.58, send to original next hop, Bob (recipient)

This network activity can also be visualized in the following diagram. Original message routes chosen by the sender are shown with solid black lines and a message on a detour is shown with a dashed line.

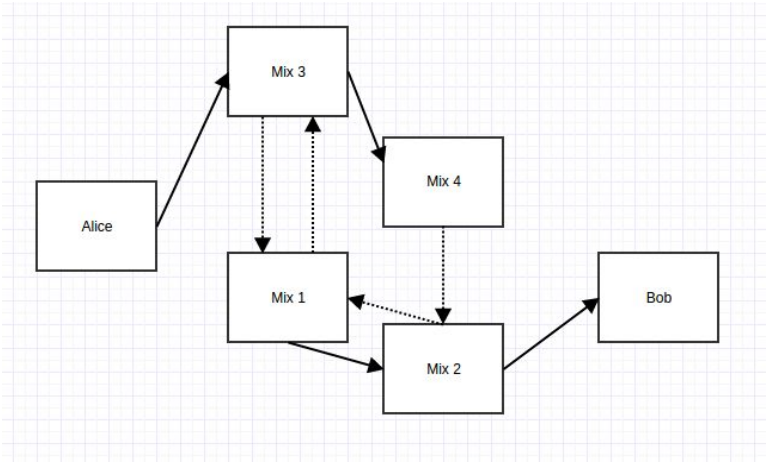

Figure 2. Actual route of message sent from Alice.

Analysis of Results

Defeating the n-1 Attack

The purpose of this modification is to defeat the n-1 attack. We will start with the worst case scenario of all mixes on the mix-net being compromised and analyze the effects of adding honest mixes one by one. We can draw conclusions based on the minimum number of mixes needed to provide anonymity.

No honest mixes

If all mixes on the mix-net are compromised, traffic analysis is not needed. Traffic can be monitored by the mixes as the message travels along its route.

One honest mix

The GAA delays the target message until the honest mix has flushed. The target message along with n-1 dummy messages with known destinations are injected. The honest mix may send the messages in the buffer to their original destinations or it may pick detours for each individual message. This is irrelevant however because the other mixes are compromised and can identify the dummy traffic. The one message that is not dummy traffic is identified as the target message.

Two honest mixes

The GAA delays and injects traffic just as it did previously. All dummy traffic is set to go to another compromised mix. The target message can either go to a compromised mix or the one other honest mix on the network. If the target message is not sent to a compromised mix the only other member it could be sent to is the honest mix. Whether or not the dummy traffic is sent to the original next hop or on a detour is irrelevant because of this identification of the target message.

Three honest mixes

The GAA delays and injects traffic just as it did previously. All dummy traffic is set to go to another compromised mix. For both the target message and the dummy traffic the honest mix will either follow this next hop or detour the traffic. As long as the target message is sent to one of the other honest mixes along with at least one dummy message, the GAA cannot identify the target message's route. This concept is visualized in the following diagram.

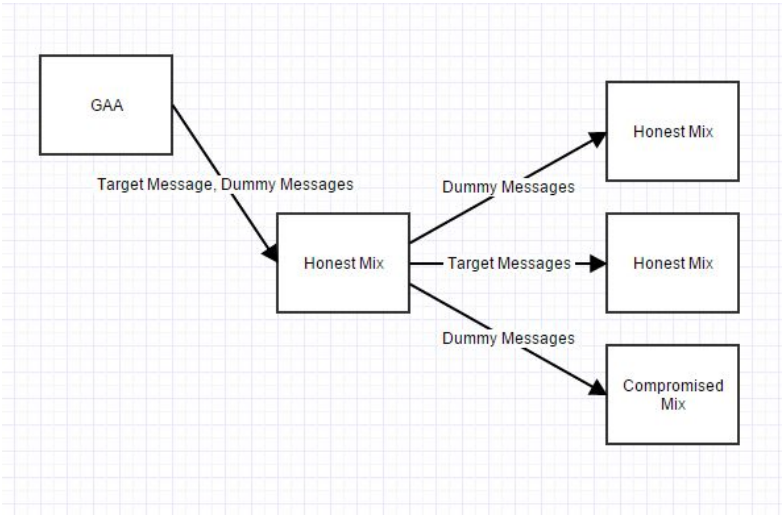

Figure 3. Three honest mixes confounding a GAA

Increasing Anonymity

As a general rule, the more mixes that are used, the greater the anonymity. This mix-net modification guarantees that a message chosen to be routed with *x* mixes by the sender will always be sent with at least *x* mixes, most likely more as the message is sent on detours. This can be helpful especially if a sender is not aware of all mixes on the network yet the mixes are.

Downsides

The only downsides of implementing loose routing in a mix-net are increased delays. The delays depend on overall network traffic and the probability that a detour will be chosen on each mix. That being said, a small increase in delays is not a significant cost for increasing anonymity.

Conclusion

Mix-nets with a threshold flushing algorithm are especially vulnerable to n-1 attacks. My goal with this project was to modify an existing mix-net design to add loose routing. This loose routing design was inspired by the design of crowds. This loose routing modification would prevent *n-1* attacks or, at the very least, make the attack more difficult for a GAA. The resulting

network design does increase anonymity at the small cost of adding delays. In my analysis I have concluded that the minimum number of honest mixes needed to prevent an *n-1* attack is three honest mixes.

Further Design Goals

My future goals for this project include adding padding to the encrypted message, adding recipient response ability, and replacing the text message with real HTTP traffic.

Every time a layer of encryption is added to the message, the message grows in size. A global passive adversary could observe the shrinking of the message as it travels over the network and is slowly unencrypted. It can use this observation to guess roughly how close the message is to its destination. To prevent this, padding can be added so that a message is always a constant size no matter where it is in its route. This padding must be randomly generated so that a passive adversary cannot easily identify it as padding. This feature could be added to my code's decryption algorithm.

Adding response ability would require further modification to the code. I would be required to design a system that includes the sender's address in the message to the recipient or encrypts a record of the route the message has taken. I would first research how Tor manages message responses and design my full implementation using what I had learned.

Replacing the text messages being transferred with real HTTP traffic will first require that I implement the response system previously discussed. This is required because HTTP traffic is bidirectional. Then the code will have to be modified to intercept the HTTP traffic from a web browser. Possible implementations involve creating a local proxy server and directing all web traffic to it or a browser extension that redirects traffic over the mix-net.

References

Chaum, David. "Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms." Communications of the ACM (1981): 84-88. Web.

Serjantov, Andrei, Roger Dingledine, and Paul Syverson. "From a Trickle to a Flood: Active Attacks on Several Mix Types." (2002): n. pag. The Free Haven Project. Web.

Newman-Wolfe, Richard. "High Level Prevention of Traffic Analysis." Computer Security Applications Conference (1991): 102-09. Web.