Increasing Anonymity in Crowds via Dummy Jondos

Benjamin Winninger

Computer and Information Science and Engineering Dept. (Undergraduate)

University of Florida

Gainesville, FL – 32611

BWinninger@ufl.edu

Abstract

In this paper I will be considering the anonymity providing system known as a "crowd", as described in "Crowds: Anonymity for Web Transactions" by Michael Reiter and Aviel Rubin. The crowd shows a great deal of promise in terms of sender and receiver anonymity against local adversaries. I will be analyzing whether the addition of a mechanism that I call "dummy jondos" can augment the crowd system, and if so, determining the advantages of using it. I will describe the development of the concept of "dummy jondos" and consider three common possible attacks on crowds: the local eavesdropper attack, the end server attack, and the collaborating jondo attack while comparing the new crowd scheme to the "traditional" crowd scheme.

1. Introduction

A crowd is a mechanism designed to protect a participant's anonymity on the Internet. In a crowd, anonymity is afforded via the uncertainty that the participants in the crowd provide. Each participant (individual user) is represented by a process named "jondo", and so participant(s) are hereafter referred to as jondo(s) in this paper. The notion of a crowd is drawn from the physical parallel of "blending into a crowd", whereby an individual (not necessarily a nefarious one) can't be easily identified among the mass of faces and bodies within the crowd.

As with its physical counterpart, the technical crowd provides progressively more anonymity as the size of the crowd grows. A man walking by himself (in essence, a crowd of size one if somebody is generous enough to call it so) or even in a group of a few people down a suburban street intuitively provides less anonymity than a man walking through Times Square around lunchtime. It is easier to recognize a person of interest when the pool of people somebody would have to search through is smaller. A jondo in a crowd is afforded anonymity by passing their (web traffic) messages to random other jondos in the crowd, and the second jondo will "flip" a biased coin to determine whether to

forward the message again to a third jondo (with probability p_f from the coin) or to send the message to the end server (with probability $1-p_f$ from the coin).

The anonymity "brains" behind the crowd lies in the fact that each jondo in the crowd is theoretically equally likely to be the originator of a message, and so if *n* is the number of jondos in the crowd, then the user's jondo will only submit the message with probability (1/*n*). From this, it should be apparent that as the number of jondos in the crowd increases, so too does the anonymity that the crowd provides. Indeed, as *n* approaches infinity the probability of an attacker getting lucky and determining which jondo sent the message approaches zero.

The greatest strength of a crowd, which is the disjoint set of users all volunteering to provide and depending on each other for anonymity, is also a weakness that should not be overlooked. For example, how many people can stay awake and active on their computers for 24 hours a day, 7 days a week? A computer scientist is probably the closest that humans come to such a feat, but even the strongest among us needs to sleep. The point is, that when a user who would otherwise be increasing the anonymity in a crowd is no longer physically present (e.g. they are sleeping from 12:00 A.M. to 8:00 A.M.), what happens to their jondo? Assuming that user is like most others, chances are that their activity will drop off, which could essentially mean that anonymity is at its lowest late at night and so an attacker can potentially gain a significant amount of information.

Is it possible to guard against the loss of anonymity when "true" crowd size isn't particularly large? It seems unfair that the night owl should be less well protected than the early bird in the example outlined above. When it comes to crowd, size is everything and so if the jondos knew that they could be equally well protected regardless of outside circumstances like human metabolic necessities then they would be much more untroubled. This is where the concept of the "dummy" jondo originated from, the idea that a crowd's appeal shouldn't be squandered by a lack of "active" jondos.

2. Goals

2.1 Anonymity

The primary goal of this research is to determine whether the inclusion of dummy jondos in a crowd will increase its resilience against three primary attacks by an adversary: the local eavesdropper attack, the end server attack and the collaborating jondos attack. These attacks will be discussed further in later sections of this paper, but the overarching idea is that increasing the size of the crowd by keeping "dummy" jondos active will maximize *n* and thereby minimize the probability that senders or receivers will be exposed.

This is to say that this research is dedicated to the degrees of anonymity provided by crowds. More accurately, this research is dedicated to observing how dummy jondos affect the probabilities of sender and receiver degrees of anonymity for active jondos. Dummy jondos will not necessarily affect the proven degrees of anonymity that exist for crowds, but instead will assure the highest possible probability for these degrees of anonymity since the guarantees hold asymptotically as *n* approaches infinity.

Table 1. Anonymity properties provided by Crowds

Attacker	Sender anonymity	Receiver anonymity
local eavesdropper	exposed	$P(\text{beyond suspicion}) \underset{n\to\infty}{\longrightarrow} 1$
c collaborating members, $n \geq \frac{p_f}{p_f - 1/2}(c+1)$	**probable innocence** $P(\text{absolute privacy}) \underset{n\to\infty}{\longrightarrow} 1$	$P(\text{absolute privacy}) \underset{n\to\infty}{\longrightarrow} 1$
end server	**beyond suspicion**	N/A

2.2 Latency

It might seem obvious that in order to guarantee a high degree of anonymity, we should just flood the crowd with an "infinite" number of jondos. While it is true that a large size for a crowd is preferable and is indeed one of the inspirations for this research, we must take care to ensure that the latency (i.e. the time it takes to send and receive a message) is maintainable. Users expect Internet actions to be fast, and so delays that might be acceptable for applications like email (where a Chaum mix might be more appropriate) are not acceptable in crowds. It is possible in some Web contexts that low latency might be a requirement, such as online stock trading where a difference of a few milliseconds induced by a poor choice of path in the crowd may lose (or gain) a significant sum of money.

While slow response times from servers is typically out of a user's control due to reasons such as high traffic load, weather interference, etc. we may be able to limit the latency within the crowd. It is simply not possible to have low latency and high anonymity because for the highest anonymity one would have a very long path that the message travels through, but for the lowest latency a path of length one would be ideal. Thus, it is necessary to examine the biased coin function and possibly implement some other restrictions with the understanding that certain applications may favor anonymity where others need low latency and so we will be attempting to find the "sweet spot" between the two if one exists.

2.3 What Dummy Jondos Do Not Achieve

As with the standard crowd, a crowd that uses dummy jondos makes no effort to prevent denial-of-service attacks. A rogue crowd member can still drop messages that it should be forwarding or can substitute wrong information to forward along the path. Dummy jondos are meant to bolster a crowd's anonymity, not serve as a cure-all for a crowd's vulnerabilities.

3. Brief Technical Crowd Overview

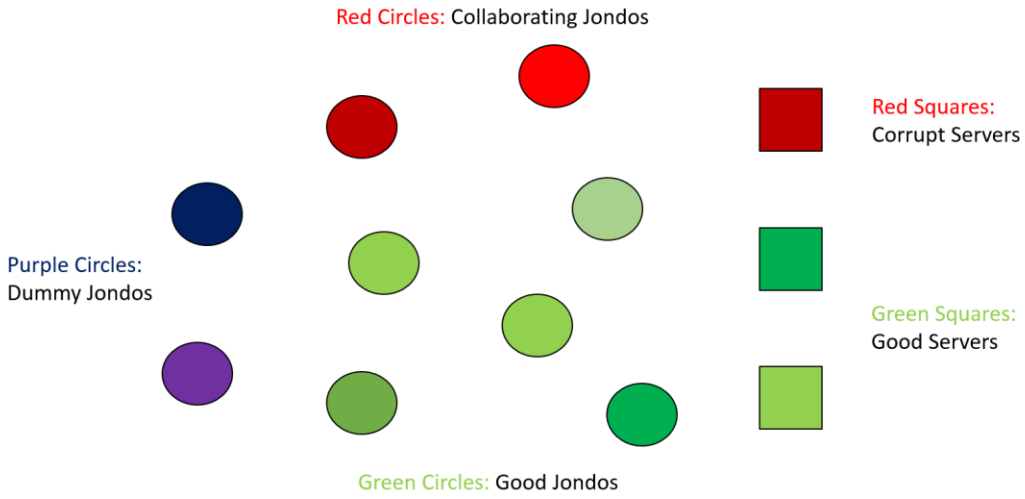
For those unfamiliar with the concepts presented in "Crowds: Anonymity for Web Transactions" by Michael Reiter and Aviel Rubin, I will try and present a quick overview however I highly recommend reading the original article.

A crowd can be thought of as a collection of users who are each represented by their respective jondo processes on their computers. When the jondo process is started, it contacts a “blender” server to request admittance to the crowd. This blender server registers the jondo’s account containing its IP address, port, and pseudonym/account name and adds it to the existing list with the other registered jondos. The blender also generates shared keys for data encryption and crowd member authentication.

When a jondo makes its first web request, it will establish a path by picking a jondo at random from the crowd members (including itself possibly) and then forwards the request to that jondo. This series of forwarding from jondo to jondo continues until the “end” jondo forwards the request to the end server. The path that was created is a virtual circuit and now all communication to and from the server goes along the forward and backward routes.

Voila! If you didn’t before, you now have a basic understanding of how crowds operate. I will now cover the concept of dummy jondos and how they help resist the attacks mentioned above, which are: Local Eavesdropper, End Server and Collaborating Jondos.

4. Dummy Jondo Concept Development and Definition



I have mentioned dummy jondos, and while I might have sold you on the idea of improving crowd anonymity, I should attempt to demystify exactly how they are set up or will function. Initially, I considered that a crowd “organizer” or initial member could have a range of proxies or otherwise spoofed IP addresses that he fed to the blender, and these dummy jondos would send requests that made them seem like normal users. This is an admirable concept, but there are two main flaws that I stumbled across while pondering how one would even go about implementing it.

The first problem is how the organizer would go about registering the proxies with the blender server. Each jondo in the crowd is associated with an account, and if there are random proxies that are

functioning as jondos, how could one possibly expect to properly setup and make these proxies seem humanlike? Not to mention that IP addresses of proxies can change or die off, and one could accidentally end up performing a denial-of-service attack on their own crowd if unresponsive jondos are not properly dispatched. Even worse, as a peer of mine pointed out: a nefarious organizer in control of all of the dummy jondos is in the perfect position to easily execute the collaborating jondos attack!

So rather than introducing outside "users" to the crowd, the concept of the dummy jondo is instead better defined by adapting the existing jondo process. I convert the typical jondo to a daemon (or non-Unix equivalent) process that will be run on startup since it should be active for the entire time that a user's computer is on. There are two modes that the jondo can be running in: "dummy mode" (i.e. the user is inactive and the jondo is sending dummy requests), and "user mode" (i.e. the user is active and the jondo is sending the true requests). The jondo will be running in dummy mode from the time the computer is booted until the time the user sends his or her first request out, at which point the jondo will transition to user mode. After a jondo in user mode goes within a certain random percentage of a user defined timeout, the jondo will switch from user mode back to dummy mode. Starting the jondo in dummy mode is important, because it is entirely possible (although perhaps unlikely considering the time we live in) that a user might not use a web browser for the duration of their activity.

For example, a user could be working on writing an essay and thus only using a text editor, and another user could be coding an application, and still another could be fiddling with a virtual machine. The point is that if these three members were registered with a crowd but had not yet sent a request, their jondos wouldn't exist anywhere in the crowd, which leads to a smaller active crowd size than theoretically possible. Recall that one of our goals is to keep *n*, the number of jondos in the crowd as large as possible to try and mitigate several attacks. In a traditional crowd, a jondo's path is established when a user sends a request, so this naturally leads to the question what exactly does a dummy jondo do or appear to do?

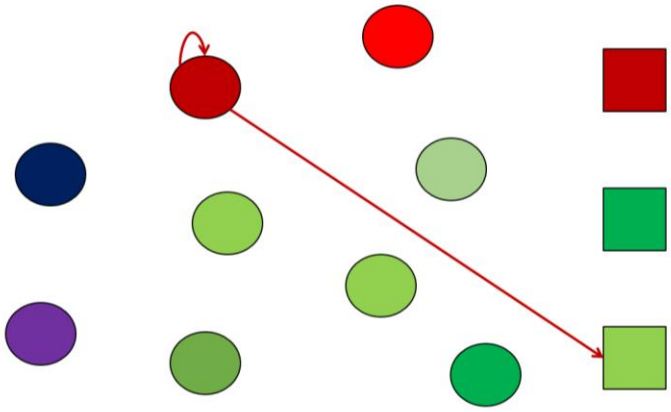
A dummy jondo should theoretically mimic what a user jondo does. Initially I considered saving a number of the user jondo's last requests, and making the dummy jondo perform these requests again. After all, what better way to make a program look like a user than to make it literally do what the user does? There are two main problems with this course of action for dummy jondos though. The first is that a user might not want the request to be repeated, which is especially true if the user just did something like donate money or purchase a product. If the dummy jondo repeats a request like this, the user will be very confused when he checks his bank account or bitcoin wallet and finds that all of his money was spent on something that he only wanted to buy one of. The second problem is a consideration of preserving anonymity, because if a user's successfully sends an anonymous request, but the dummy jondo keeps sending this request, it could ruin the existing anonymity for that request.

Instead, we create a list of common default servers like Google, Facebook, YouTube, etc. to perform a set of commands on including ping, whois, and nslookup among others. Of course, the user would be able to edit both of these lists in a config file should he so choose. Now, the dummy jondo performs a random command on a random server in a bounded random time interval. The dummy jondo can even drop the response messages from the server since the user doesn't care about them. The obvious advantage is that communication with a Google server, should it be observed by an attacker, wouldn't be particularly informative considering that it is the most visited website in America.

The randomness helps make the program seem more human-like since everybody's browsing habits are different and although there exist some statistics and research on the subject, I will assume that this simple randomness scheme is sufficient.

In the same vein of randomness, I have considered the possibility of reorganizing the crowd every time a jondo transitions from user mode to dummy mode. The reason behind the consideration is that adding the capability for the crowd's layout to drastically change at any given moment adds a dynamic element to the environment and makes intersection-like attacks more difficult. A new path will be generated for the now-dummy jondo and all paths that included the jondo when it was in user mode will have to change at least one link. An optimist might say that rearranging the crowd could even disrupt a collaborating jondo attack, but it is also possible that the new paths might set the "good" jondos up for a different collaborating jondo attack. I am still ambivalent on the topic and will continue to analyze it, because other issues pop up like what happens to the shared keys between pairs of jondos and what happens when these switches occur so often that they start to disrupt normal traffic from other active users.

5. Local Eavesdropper Attack



A local eavesdropper is an attacker who can observe any sender traffic from a particular user's computer. A *local* eavesdropper is local because it can't see any traffic from other user's excepting the one that it is listening to. Recall from Table 1 above, the sender's anonymity is "exposed" for a local eavesdropper. If a jondo sends a request out without receiving a request in, it is fairly obvious who is the sender, and crowds make no effort to protect against this since typically the only thing that a local eavesdropper will see is an encrypted address for the next jondo.

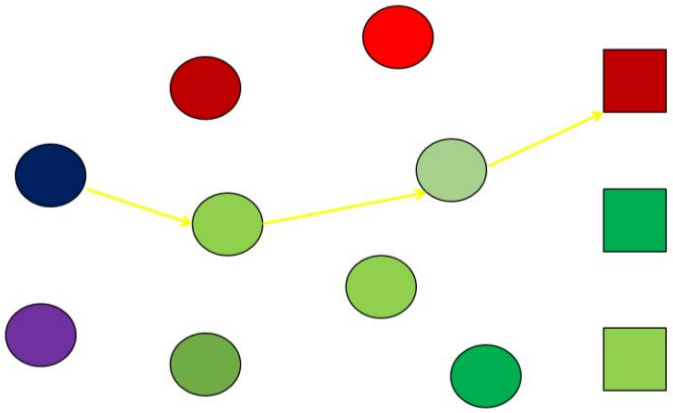
Notice that I said *typically* because if the jondo is particularly unlucky, then it will submit its request to the end server by itself (i.e. without any intermediate jondos serving as proxies)! In my initial consideration of how a dummy jondo defined, including an arbitrary number of new jondos could all but nullify the local eavesdropper attack. Recall that as n increases towards infinity, $1/n$ will drop off to zero so a local eavesdropper will have virtually no chance of finding a good target to listen to. However, in

our redefined scheme we can't increase the potential size of our crowd to a large number that we see fit.

What we do achieve with the new scheme is get as high of an *n* as we possibly can relative to the size of the crowd. What this means is that since there is a bounded size of registered users in the crowd, we try to get the largest value for *n* less than or equal to that size. Essentially, this is no different than how the traditional crowd attempts to combat the local eavesdropper. The main difference is that while the traditional crowd is content with only having active users as members, the new scheme strives to bring in as many members as possible, in effect maximizing the percentage of available users/registered users.

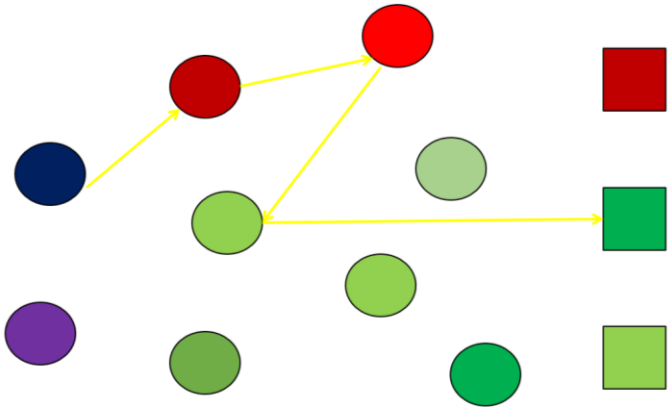
What is the difference you might ask? We already know that the probability of the attacker getting lucky and witnessing a jondo send its request to the end server is 1/*n*, where *n* is the size of the crowd. Both the traditional scheme and the new scheme will have *n* as the size of the crowd, but in the new scheme we attempt to bring in *d* dummy jondos, who would otherwise not be members of the crowd. Thus the new scheme's probability of the attacker getting lucky is 1/(*n*+*d*), which is obviously a value less than or equal to 1/*n*. This improvement in probability scales with *d*, so the difference could be something minimal (although not insignificant) like 1/10 vs 1/11, or it could be massive like 1/10 vs 1/20! Regardless, the new scheme guarantees to be no worse than the traditional crowd's degree of anonymity offered for resisting a local eavesdropper.

6. End Server Attack



The end server attack occurs when the web server to which a jondo's request is sent is compromised. As the receiver of the request is compromised, there is no receiver anonymity possible! However, the sender anonymity is preserved exceptionally well provided that there is more than one member in the crowd. The sender anonymity for the end server attack is guaranteed to be beyond suspicion independent of path length and number of members in the crowd. The reason is that any sender is equally likely to have been the initiator of a request, and the end server can't know if it was the last sender is the initiator or merely a forwarding jondo. As a result, the analysis of this attack is severely limited and uninteresting. The new scheme of using dummy jondos provides exactly the same degree of anonymity offered for resisting the end server attack as the traditional scheme.

7. Collaborating Jondo Attack



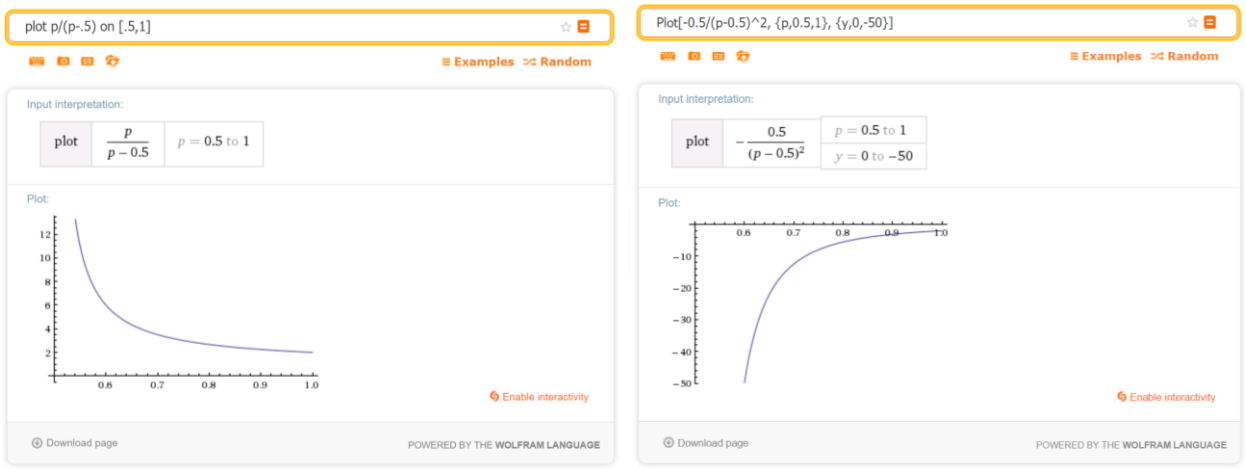
A collaborating jondo attack takes place when a group of crowd members that can view all communication to and from "collaborating" users and pool this information together to find the sender of a request. Despite not being the greatest overall threat because of the local eavesdropper's capabilities, this attack poses the greatest consistent threat to the anonymity of a crowd. As such, a worthwhile anonymity improvement to a crowd is one that helps resist against collaborating jondos. In order to be afforded probable innocence, the number of members in the crowd $n >= (p_f*(c+1))/(p_f-(1/2))$ where p_f is the probability of forwarding to another jondo and c is the number of collaborators in the crowd. There are therefore two things that we can do in order to increase the chance of satisfying the above inequality assuming a constant number of collaborators: we can increase n, or we can minimize $p_f/(p_f-(1/2))$.

In terms of increasing n, I have described this in the section on the local eavesdropper attack. Where a traditional crowd scheme is represented by the aforementioned formula, the new scheme can be represented by a similar formula: $(n+d) >= (p_f*(c+1))/(p_f-(1/2))$ with d representing the number of dummy jondos as members. Here we can say the same of defending against collaborating jondos that we did for the local eavesdropper attack: the new scheme guarantees to be no worse than the traditional crowd's degree of anonymity offered. In the outdated description of a dummy jondo, ideally the dummies could be pumped into the crowd until the number of collaborators c became prohibitively high. Alas, things can't always be so simple. Is there a way to improve on this newly defined formula though?

We could attempt to find a "minimum" for the expression $p_f/(p_f-(1/2))$. I have put minimum in quotation marks because there is no true mathematical minimum for this expression, so we instead turn to locating where the lowest p_f falls that offers a value relatively close to zero and also for which a neighboring value isn't worth using. Below are the plots of this expression and its derivative. It is impossible to say without knowing the needs of the crowd what the "proper" value of p_f should be even after analyzing these graphs. It appears that the "sweet spot" for minimizing the impact of $p_f/(p_f-(1/2))$ while considering latency is somewhere in the range of [0.75,0.85], but as the size of the crowd grows

this could mean substantial latencies when collaborating jondos might not even be capable of succeeding.

For this reason, it might be desirable to have a sliding scale type of function for the p_f in which the value is initially somewhere in the range of [0.75,0.85], but then drops slightly over time to accommodate growing crowd size. If for some reason a significant amount of members leave the crowd, p_f would increase to accommodate that. I believe that a simple linear function should work well enough here, there is no need to do anything much more complex. For instance, if I choose an initial value of 0.85 for p_f my resulting function could look like $p_f = -0.005n + 0.85$, stopping when p_f hits a predefined value, say 0.6. Clearly, the smaller the constant multiple of n is, the slower the function will decrease the initial value of p_f. This exemplifies the tradeoff of path length (anonymity) for latency (response time from the server).



8. Conclusion and Further Work

In this paper, we reviewed the anonymity system known as a crowd as initially described by Reiter and Rubin and how we could possibly adapt it to encompass a new feature which I called dummy jondos. The concept of a dummy jondo has evolved from the organizer adding an arbitrary number of jondos to the crowd (thus setting the crowd up for an easy collaborating jondo attack) to bringing in as many registered jondos as possible (thus the maximum crowd size is less than or equal to the number of registered users). It was demonstrated that by forcing users who would otherwise not be active members in the crowd to be members in dummy mode we could achieve at least as much anonymity as provided by traditional crowds for the local eavesdropper and collaborating jondo attacks, but the end server attack remained unchanged. We have considered how a dummy jondo should act, whether jondos should dynamically rearrange in the crowd when transitions from user to dummy mode occur, how the probability of forwarding a message could help increase anonymity, and briefly considered the tradeoff of path length for this increased anonymity.

Ultimately, since this was entirely a theoretical paper I will likely need to redefine various aspects when I begin implementation of this system. I have already shown that some ideas needed to be

redefined along the development process (what a dummy jondo is, how it should act, etc.), so it is almost guaranteed that some concepts will need to be tweaked at the very least. In particular, when I implement the system in a language like C or C++, I will be able to apply tests and receive real data to either prove or refute arguments made in the discussions above. Testing things like optimal path length (and p_f) and whether the default commands for a dummy jondo are reasonable or not will pave the way for an actual deployment of this new system.

9. References

Diaz, Claudia, and Bart Preneel. "Taxonomy of Mixes and Dummy Traffic.". University of Leuven. Web. <http://freehaven.net/anonbib/cache/taxonomy-dummy.pdf>.

Newman, Richard E. "Anonymity – Crowds." CIS 6930/4930 Cryptographic Anonymity. University of Florida. Web. 22 Nov. 2015. <http://www.cise.ufl.edu/~nemo/anonymity/lectures/lect04aCrowds.ppt>.

Reiter, Michael K., and Aviel D. Rubin. "Crowds: Anonymity for Web Transactions." ACM Transactions on Information and System Security 1 1.1 (1998): 66-92. CIS 6930/4930 Cryptographic Anonymity. University of Florida. Web. 20 Nov. 2015. <http://www.cise.ufl.edu/~nemo/anonymity/papers/crowds:tissec.pdf>.

Shields, Clay, and Brian Neil Levine. "A Protocol for Anonymous Communication Over the Internet." Georgetown University. Web. 28 Nov. 2015. <http://people.cs.georgetown.edu/~clay/research/pubs/hordes-ccsfinal.pdf>.

Wright, Matthew K.; Adler, Micah; Levine, Brian Neil; and Sheilds, Clay, "The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems" (2002). Computer Science Department Faculty Publication Series. Paper 164. <http://scholarworks.umass.edu/cs_faculty_pubs/164>.