This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

MIX-Crowds, an Anonymity Scheme for File Retrieval Systems

W.H. Tang
Department of Computer Science
The University of Hong Kong
Email: whtang@cs.hku.hk

H.W. Chan
Department of Computer Science
The University of Hong Kong
Email: hwchan@cs.hku.hk

***Abstract*—In this paper, we propose an anonymous scheme for file retrieval systems, MIX-Crowds, in which it is harder for an attacker to identify the requester of the file by making use of the idea of MIX [7] and Crowds [20] to establish a path from the requester to the file holder. Result shows that predecessor attack [26] is much more difficult to succeed compared with Crowds [20]. We are able to reduce the estimated number of rounds needed for successful predecessor attack for MIX-Crowds. We also propose a file transfer strategy according to file size. With such strategy, requests for small size files can be completed faster while the downloading time of large size files only increases slightly.**

I. Introduction

Since the introduction of Internet, protection of the privacy of users has been one of the paramount concerns for the public. Users send and receive emails, surf websites for information, share ideas with others in newsgroups and message boards via the Internet. In some Internet applications, for example, online shopping, customers need to reveal their identity to the server. However on some occasions, one may wish to hide one's identity in sharing and obtaining sensitive information to and from others. Thus, a large number of anonymous applications with different purposes has been proposed over the years. Some representative examples are Crowds [20] used for anonymous web browsing; Freenet [8], Free Haven [9] used for anonymous files distribution; Babel [12] used for anonymous email; Onion Routing [10] used to anonymize TCP-based applications.

From the introduction of Napster [2], peer-to-peer file-sharing systems have been very popular. However, many common peer-to-peer file-sharing applications offer no protection of users anonymity. By examining the IP address of the peer node where a file request comes from or a file is published, it is very easy to find out the identity of the node. Privacy of users in these peer-to-peer applications would be compromised easily.

As a result, a couple of anonymous file publishing and retrieval systems were proposed. Most of these systems achieve anonymity by sending the request for a file to at least one other node instead of contacting the file holder directly. The main differences among these systems are how they locate the file and how they choose the next node in a sequence of nodes to the file holder. In some anonymous file sharing applications, like Freenet [8], nodes are assigned random node IDs. To search for a file with key k, the original requester sends out a request for the file to its neighbor with the closest node ID to k. The file request will then be forwarded by this node to its neighbor with the closest node ID to k. Subsequently, it is likely that the request will finally reach a node which currently holds the file. This scheme has three drawbacks: (i) when a node receives a request from its predecessor node, the farther the file key is from the predecessor node's ID, the more likely that the predecessor node is the original requester. (ii) as the size of the network grows, the path length can be very long [8]. This increases the time of the file transfer, as it is more likely that some of the nodes on the path leave the network during the transfer. Consequently, the transfer has to be performed again, increasing the chance that an attacker will identify the requesting node. (iii) as the network is loosely structured, there is no guarantee that a file will be located successfully, even it exists in the network. Some other applications like Ghostshare [19] use an overlay network to guarantee that file will be located successfully. The next node is usually chosen from the routing table of the requesting node. However, the attacker may change entries in a victim's routing table setting the attacker as a neighbor node, as pointed out in [6]. In that case, all of the victim's requests will be revealed by the attacker. Moreover, it is possible that the attacker can log down the predecessor of the request, known as the predecessor attack, in a fashion described in [26]. With sufficient numbers of file re-transfer, the attacker will be able to identify the file requester with high probability.

We propose a new anonymity scheme, MIX-Crowds, for file sharing network. MIX-Crowds makes use of the idea of MIX [7] and Crowds [20] to establish a path from the requester to the file holder, where predecessor attack is much more difficult to succeed compared with Crowds.

This paper is structured as follows. In Section II, we describe the system design of MIX-Crowds and how anonymous file retrieval can be achieved based on the use of an overlay network together with other system components. Other issues such as nodes management, file publishing and transfer strategies are also discussed. In Section III, anonymity and security analysis of MIX-Crowds are performed. Our analysis on anonymity is mainly based on modified version of predecessor attack. We also compare MIX-Crowds with other anonymity communication systems with regard to the degree

978-1-4244-3513-5/09/$25.00 ©2009 IEEE

of anonymity, overhead, as well as other performance and security issues. The paper is concluded with some suggestions of future research directions for MIX-Crowds in Section IV.

II. System Components of MIX-Crowds

A. Chaum's MIX and Crowds

Chaum's MIX [7] achieved anonymity by public key cryptography. By encrypting the message together with the recipient address and sending it through a series of MIXes, the correspondence between the sender and the receiver can be hidden. Due to the use of layered encryption, the MIXes will have no idea of what is being transferred. Many improvements to MIXes system have been proposed [15], [14], [5]. However, due to the high latency associated with the use of public key cryptography, batching of messages, etc, these systems can only be used in high latency systems such as emails.

Crowds [20] provides anonymity of users from the web servers. Instead of contacting a single anonymous proxy like Anonymizer [1], a request first goes through a "jondo" (node) which in turn will forward the request to another jondo with probability p before submitting the request to the web server. As a result, jondos are able to view the content of the request and reply. By adjusting p to a higher value, different degree of anonymity can be achieved. Analysis in [26] shows that a higher p value can have stronger defense against predecessor attack. For details of predecessor attack, please refer to [26].

The inefficiency of MIX is due to encryption and decryption of the whole message. It is hard for nodes on the path to determine the encrypted message. However, if all intermediate nodes do not know what file they are transferring, it will be very slow for the file to spread in the network in spite of the high demand of the file. This is because the file requester, who wishes to stay anonymous, is likely not to send the file to others. On the other hand, it is more flexible for Crowds to spread the file where the path is determined by the next node recursively, but the content is known to all nodes on the path. In MIX-Crowds, the advantages of both schemes are taken into account.

B. Network Topology

1) Directory Server: In a MIX system, the sender chooses a series of MIXes that the message will go through and wrap the message and address of MIXes together with their public keys. Thus to apply layered encryption, each node must have a public-private key pair. In MIX-Crowds, we assume that there is a directory server, like the blender in Crowds, which keeps the public key information in addition to the list of nodes. In other words, each node will know the public keys of all other nodes. Each node submits its public key to the directory server, and keeps its private key itself. The directory server keeps the public key and the corresponding IP address for each node.

2) Secure Node ID Assignment: In Chord [23], it is assumed that node IDs are assigned by applying a hash function on the IP addresses. This is not secure, as stated in [6]. An attacker holding a large number of IP addresses can easily target at a particular node in the overlay network, viewing and directing all the messages which the node sends out to the overlay network.

Some suggestions are given in [6] to prevent an attacker from registering a large number of nodes in the network: the first preventive measure is binding node IDs to real-world identities, however this would compromised anonymity . The second measure is requiring nodes to consume computing resources in order to be able to use a given node ID with an IP address, for example, finding a string x such that the result of a one-way hash function of x together with the IP address and the node ID will have certain number of bits equal to 0. However, this would increase the number of choices for node ID by computationally powerful nodes. Also this may take a long time for nodes with limited computational resources.

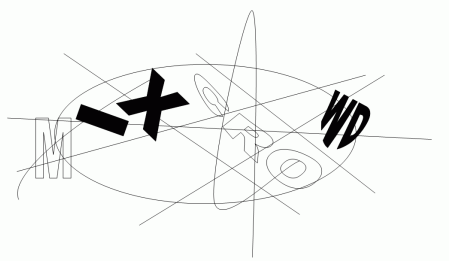

Fig. 1. Sample of CAPTCHA Challenge

It is not effective to limit the rate of an attacker joining a network by computational resources, whereas it is effective to bind node ID to a human entity. Hence we propose using CAPTCHA [25] to achieve the effect. CAPTCHA stands for "Completely Automated Public Turing test to tell Computers and Humans Apart", which can be used to determine whether the user is a human based on some challenge-response tests. For example, by asking the user to identify the characters in a distorted image which is hard to be recognized by a computer program. Yahoo! [4] uses CAPTCHA to prevent web robots from registering accounts. Figure 1 shows a sample of CAPTCHA challenge provided in The CAPTCHA project [3]. The word "MIXCROWD" has to be provided in order to pass the challenge. Some variants of CAPTCHA, such as logical and mathematical questions, are also used on the Internet.

In MIX-Crowds, CAPTCHA can be used as follows: before joining the network, a node first needs to contact a directory server, and the directory server will return a CAPTCHA challenge to the node. If the user inputs the response correctly, the directory server will accept the public key provided by the user. Then the directory server will provide a certificate for the user and update its database record. This limit the speed of the attacker joining the network, as human judgment is needed in the joining process. However, if only a single CAPTCHA challenge is required for joining, it may be possible for the attacker to use low cost labor to join the network. Thus periodic revalidation of node IDs may be necessary. Nodes can revalidate their node IDs at any time. Given the on-time for most ($> 95\%$) of the nodes in a peer-to-peer file-sharing network is less than one day [16], it is reasonable to require

a node to revalidate their node IDs in one day or two. Thus a signicant amount of human resources is needed for the attacker to maintain a large number of nodes in the network.

3) File Location Scheme: Table I summarizes the symbols used in this section.

Symbol	Definition
N_i	Node i in the network
F_i	File i in the network
ID_i	ID for N_i in the main network
$ID_{i,j}$	ID for N_i in F_j sub-network
R_i	Root node for F_i
E_i	Entrance node of F_i sub-network
H_i	Key of F_i
$H_{i,j}$	Key used to forward request for F_j from N_i at R_j
$\cdot$	Concatenation operator

TABLE I
LIST OF SYMBOLS USED IN FILE LOCATION SCHEME

In locating F_i, we need H_i which is a 160-bit SHA-1 content hash of F_i. We assume that the nodes already have the key of the file when they perform searching, as in Freenet [8]. Keyword searching may also be added to the system using the scheme proposed in [27], [19].

Chord [23] is used as the underlying overlay network with some security improvements mentioned in [6]. The improvements include protection against Sybil attack [11], bad routing table updates and dropping messages. The reason why Chord is chosen is that its routing table is tightly constrained and it has fixed node ID assignments, which is suitable for the improvement on security.

In Chord, replicas of objects (files) will be sent to replica nodes directly by the publisher. This would compromise anonymity as replica nodes will know who the publisher is. Moreover, replica nodes may leave the system, resulting in handover of the file to other nodes, which may take a long time for large files. Furthermore, a node having its ID close to the key of some popular files will likely use up a lot of its uploading capacity for the files.

Instead of using a set of replica nodes to keep the replicated file, we use a second level Chord network for each file, which will be managed by the nodes holding the file and the replica nodes.

Below we describe our proposed scheme of file location with the use of the overlay network Chord. In Figure 2, the whole Chord network consists of 8 nodes. N_2 is the root node for F_1 held by N_1 and N_7. For simplicity, the replica nodes for F_1 is not shown. Instead of keeping references to peer nodes holding F_1, N_2 only keeps a pointer to N_7. Note that we can keep several pointers for fault tolerance. When N_2 receives a request for file F_1, it forwards the message to the F_1 sub-network through N_7.

Assignment of node IDs in the F_1 sub-network depends on H_1 and ID of nodes holding F_1 (N_1 and N_7). In general, file sub-network node IDs will be assigned as follow ($\cdot$ stands for concatenation):

$$ID_{i,j} = \text{SHA1}(ID_i \cdot H_j)$$

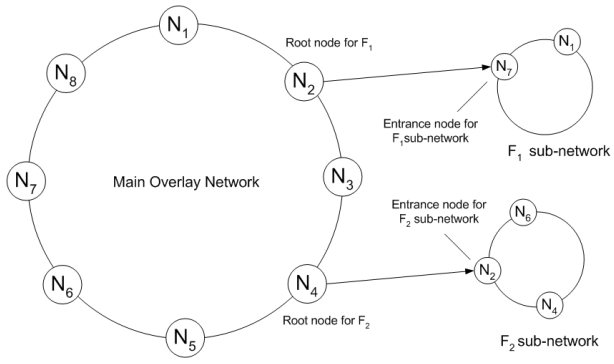

Fig. 2. The overlay network structure of MIX-Crowds

The key $H_{i,j}$ used by R_j(the root node of F_j) to forward request from N_i in the F_j sub-network will be calculated as follows:

$$H_{i,j} = \text{SHA1}(ID_i \cdot H_j)$$

The calculation of $H_{i,j}$ is almost the same as that of $ID_{i,j}$, the only difference is that N_i is sending a request for F_j instead of being a holder of F_j.

Network/Sub-network	File Key	ID
Main	-	ID_2
F_2 sub-network	H_2	$ID_{2,2} = \text{SHA1}(ID_2 \cdot H_2)$

TABLE II
IDS HELD BY N_2

File	File Key	Entrance Node	ID
F_1	H_1	N_7	ID_7

TABLE III
ENTRANCE NODE TABLE FOR N_2

Table II shows the IDs of N_2 in the main chord network (i.e. ID_2) and the file sub-network (i.e. $ID_{2,2}$). Table III shows the entrance to file sub-network kept by N_2 (only F_1 in this case). The forwarding tables for the chord networks are not shown.

Now suppose N_7 requests file F_2. First in the main chord network it sends the request to N_4 (which is root node of F_2). N_4 will then send a request message for F_2 with the key $H_{7,2} = \text{SHA1}(ID_7, H_2)$ to N_2, which is the entrance node (E_2) kept by N_4 for the F_2 sub-network, N_4 will ask N_2 to forward the request further to the node with "closest" ID to $H_{7,2}$ in F_2 sub-network, which is N_6. Finally N_6 sends F_2 back to N_7.

C. Anonymous File Retrieval

In common anonymous file retrieval system (referring to table IV), when N_R wants to obtain a file F, it contacts the file holder N_H using a Crowds like scheme: to reach the file holder, the initiator N_R forwards the request R_F containing the file key H_F of file F to another node N_1 first. N_1 will submit the request to the overlay network with probability

Symbol	Definition
F	The file to be requested
H_F	Key (Hash Value) of the file F to be requested
R_F	Request for the file F to be sent
N_R	Requester of the file
N_H	Holder of the file
N_P	Publisher of the file
N_i	Node i in the network
K_i	Public Key of N_i
K_i^{-1}	Private Key of N_i
S_i	Temporary Symmetric Key generated by N_i and used by N_{i+2} to encrypt the file
A_i	IP Address of N_i

TABLE IV
LIST OF SYMBOLS USED

$1-p$, or forward the request to another random node N_2 with probability p. N_2 repeats the forementioned procedure performed by N_1. Finally, the request will reach N_H through the overlay network, and F will be returned to N_1 through the original path, in reverse order, as illustrated in figure 3.

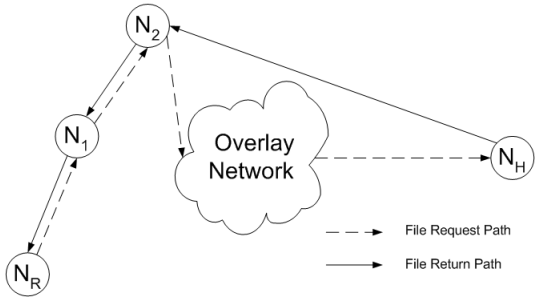

Fig. 3. Anonymous file transfer using MIX-Crowds

If we use the above scheme to reach N_H, there are drawbacks. Each intermediate node observing a request for F will know that its predecessor has a certain chance to be N_R. If there are 10% of attackers in the network, there is 10% chance that the file requester will be logged. If $p = 0.5$, each intermediate node can be about 50% sure that the predecessor is the file requester for F. Moreover, if the transfer is long enough such that some of the intermediate nodes leave the system, the transfer path have to be reset. If the number of reset becomes high, it is likely that the file requester will be revealed.

To minimize the successful chance of this type of attack, we encrypt the request with more than one public key. To request F, N_R first chooses more than one node (e.g., the two nodes N_1 and N_2 in figure 4), and construct a message M as follows:

$$M = K_1(A_2, K_2(R_F, S_R)) \quad (1)$$

where S_R is the symmetric key generated by N_R to be used by N_2 to encrypt F to prevent N_1 from knowing what file request is being transferred. If AES is used to encrypt the file, performance should not be a problem. It takes about 30 clock cycles per byte for encryption and decryption using different implementations of AES on Pentium II machines [22]. Thus the time for encryption and decryption should be small compare to the transfer delay for the file. When N_1 receives M and apply K_1^{-1} on it. It only reveals A_2 and another encrypted message. Only N_2 will know about the content in R_F. However, N_2 does not know who is the predecessor of the request (i.e., N_R). N_2 will submit the request to the overlay network with probability $1-p$. With probability p, it chooses two other nodes N_3 and N_4, creating the message $M' = K_3(A_4, K_4(R_F, S_2))$ and send it to N_3. The process repeats until the request is submitted to the overlay network.

When the request reaches the file holder N_H, N_H returns the file back via the original path to N_R. The above process is illustrated in figure 4.

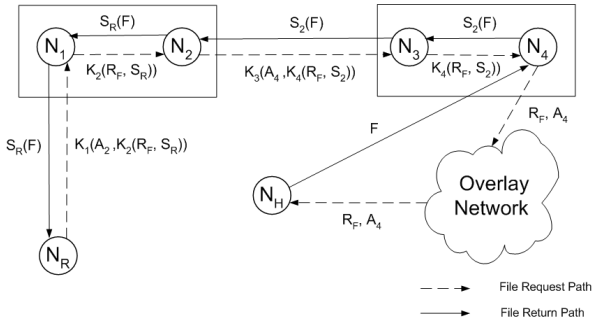

Fig. 4. File transfer using MIX-Crowds

D. Delegate Nodes for File Publishing

If anonymity is needed for the file holders, delegate nodes can be used so that no actual file holders will register themselves on the network. The idea of delegate nodes works as follows: first the file publisher or a file holder N_H holding F chooses a random node on the network as its delegate node N'_H. At that time, N'_H does not have the file F. The delegate node pretends it to be N_H and register itself on the sub-network. When a request reaches N'_H, it directs the request to N_H. After that, N_H returns the file to N'_H and then to the requesting node. N'_H may further choose another delegate node to increase the anonymity for the file holder N_H. After the transfer, the number of file holders will increase (explained in later sections), N_H can stop being a file holder and N'_H can leave the sub-network, minimizing the risk of being accounted for distributing the file. As a result, it is difficult for a collaborator to identify all the nodes holding and distributing the file currently using the overlay network.

E. Optimizing File Transfer

1) Four Classes of File Transfer: Study in [13] shows that most requests in a peer-to-peer file sharing network are for "small" objects. 90% of file transfers are smaller than 10MB in file size. However, they only occupy about 15% of the total traffic. On the other hand, less than 10% of file transfers are over 100MB, but accounting for about 65% of the total traffic. This implies that a small percentage of large file requests will greatly jeopardize the transfer and queuing

time of typical small file requests if the file size is not taken into consideration.

In order to shorten the transfer time of the majority of the files in MIX-Crowds, we separate files into 4 classes with respect to their sizes: $0-2$MB, $2-10$MB, $10-100$MB, >100MB. This is based on measurements in [13] where file size is separated into three prominent regions ($0-2$MB and $2-10$MB is combined into one class). The reasons of separating files <10MB in size into two classes are: (1) the regions $0.1-2$ and $2-10$ are of different slopes (quite significantly) (2) typical sizes for text documents, images as well as description files which can be used for keyword searching are less than 2MB.

Most of the nodes on the Internet have uploading capacity much less than their downloading capacity. Therefore we focus on how the uploading capacity is to be allocated to different classes of file transfer. A fraction of uploading capacity will be reserved for each class of file transfer in each node. If a node has 40KB/s of uploading capacity to be evenly allocated among four classes, 10KB/s of uploading capacity will be reserved for each class. When all the transfers in a particular class are completed, the uploading capacity will be temporary allocated to other classes. In this case, transfers of small files can always be done in a relatively short time. Moreover, file transfer of large files can also be performed at a reasonable speed. Overall, the average transfer and queuing time can be shortened.

2) Limitation on Maximum Number of Connections: In most source rewriting anonymous file retrieval systems, a file will propagate through several nodes before reaching the requester. In this case, when a file is transfered, the file transfer time depends on the connection with the lowest transfer rate. If we limit the maximum number of connection such that each connection can have a minimum guaranteed uploading capacity, the transfer time can be reduced. For example, if the minimum guaranteed uploading capacity per connection is $5KB/s$, then the maximum number of connections is 8 for a node with an uploading capacity of $40KB/s$. As the average transfer time is shorten, it will be less likely that the connection will be broken during the transfer. As a result, the chance that a node needs to reissue its request is reduced.

As mentioned in section II-E1, uploading capacity will be shared evenly among all classes of file transfers. Therefore if a node has an uploading capacity of $40KB/s$ and a minimum uploading capacity $5KB/s$ per connection then the maximum number of connections is 8 and the maximum number of connections per class is 2.

Simulation results using the aforementioned file transfer strategy in enhancing the performance of MIX-Crowds are given in [24].

F. Selecting the Next Two Nodes

Before asking other nodes to help with the transfer, a node will first determine if other nodes are currently overloaded (having a lot of requests in the transfer queue). Instead of using a global server keeping track of workloads of the peers, result in [28] shows that the throughput of peer-to-peer file sharing systems using the following selection algorithm is as good as other sophisticated or global algorithms. Thus we can make use of a node selection algorithm described below:

1) The requesting node N_R randomly chooses two nodes N_1 and N_2 first. N_R then sends a different list of node addresses (randomly selected from the address list given by the directory server, with no repeating entries) to N_1 and N_2. N_1 and N_2 will then ask for the current load value (e.g., the number of requests in the queue divided by the number of available connections) of each given node in the list provided by N_R. For example, node N_3 and node N_4 which have the lowest load values in the list will be returned to N_R as a result.
2) N_R can then issue a file request to N_3 and N_4.

In this way, we can ask for the current individual load value of other nodes while not compromising anonymity much. The reasons are as follows:

1) If N_R asks the load value of each node in the list directly, the probed node may be able to guess who is the requesting node (by referring to the most frequent peer(s) asking for the node's load value). Also a compromised node can always return a very low load value to N_R in order to increase the chance of being selected.
2) If N_1 or N_2 is compromised, it cannot control the list of nodes selected by N_R (unless the directory server is compromised). N_1 and N_2 can only ignore N_R's list of nodes.
3) For the nodes in each list (selected by N_R), they don't know which node is asking for their load values, unless they compromised with N_1 or N_2. Reporting a low load value indicates that the node is able to serve for the request in a short time. Therefore tracking of dishonest nodes may be needed with the help of a reputation system [21].

In this section, we have introduced the system components of MIX-Crowds and the scheme on how anonymous file retrieval can be done. With the help of a directory server, public key information can be used to hide the content of the file from the next node. File location can be done with the use of an overlay network. Some enhancements to the system such as dividing file transfers into classes, limiting the maximum number of connections, together with a node selection algorithm taking load balancing into account have also been made.

III. Anonymity Analysis of MIX-Crowds

In this section, we will perform anonymity and security analysis on MIX-Crowds. We will first analyze the anonymity of MIX-Crowds based on [26] with some modifications and extension. Then we will compare the anonymity of MIX-Crowds with other anonymous communication systems. Finally, security analysis under some common attacks will be provided.

A. Predecessor Attack for the Proposed Scheme

In [26], predecessor attack on Crowds [20] is defined as follows: during path formation, the collaborator logs down the predecessor of each request it receives. After a number of rounds, the chance that the initiator will appear is much higher than that of the non-initiator. Thus identification of the initiator is possible in a long term.

In our proposed scheme, to identify which node is the file requester of a particular file F, the collaborator must identify two entities: the file being transferred and who is the predecessor. Suppose N_R sends a request message $M = K_1(A_2, K_2(R_F, S_R))$ to N_1. If N_1 is a collaborating node, it does not know which file is requested by N_R. If N_2 is a collaborating node, it also does not know which node is requesting F. The collaborator will know that N_R is requesting F if N_1 and N_2 are both controlled by the collaborator. However, the collaborator cannot be sure that N_R is the file requester since N_R may be forwarding request on behalf of other nodes. There is a method that the collaborator can increase its chance of identifying the file requester. If one of the nodes selected by N_R is compromised, the collaborator can simply ignore the request, hoping that N_R will next choose two collaborating nodes.

Suppose there are n nodes in the network, with c collaborating nodes. Denote $C_{i,j}$ as the event that i nodes out of j nodes chosen by N_R is compromised. The chance that both nodes chosen by N_R are compromised is (assume that the network is large enough so that the effect of node replacement is negligible):

$$P(C_{2,2}) = \left(\frac{c}{n}\right)\left(\frac{c}{n}\right) = \frac{c^2}{n^2} \tag{2}$$

The chance that one of the two nodes is compromised is:

$$P(C_{1,2}) = \binom{2}{1}\left(\frac{c}{n}\right)\left(\frac{n-c}{n}\right) = \frac{2c(n-c)}{n^2} \tag{3}$$

In Crowds, H_k denotes the event that the first collaborator on the path occupies the kth position on the path, where the initiator occupies the 0th position and $H_{k+} = H_k \vee H_{k+1} \vee ....$ In our scheme, two nodes are chosen in a group. Thus we denote H_k as the event that the first pair of collaborators on the path occupies the kth pair position on the path. For example, H_1 denotes the event that the all nodes chosen by N_R are compromised. Also denote I the event that the first group of collaborators on the path is immediately preceded by the file requester.

It is obvious that $P(I) = \frac{c^2}{n^2}$. However, if the collaborator uses the strategy which ignores the file request unless both nodes chosen by N_R are compromised, it can increase its chance of identifying the predecessor. Denote $P(I')$ be the probability of identifying the predecessor using such strategy.

$$\begin{aligned} P(I') &= P(I) + P(C_{1,2})P(I') \\ P(I') &= \frac{P(I)}{1 - P(C_{1,2})} \end{aligned} \tag{4}$$

For example, when $n = 1000$ and c = 100, $P(C_{1,2}) = 0.18$, $P(I) = 0.01$ and $P(I') = 0.0122$. We can see that the chance is increased by about 22%. Our analysis follows similarly to that in [26] but with the reduction of number of rounds needed in identifying the requester.

The justification of the above equation is reasoned as follow. At first, if the first two nodes chosen by the file requester are both collaborating nodes, the collaborator succeeds. Otherwise, if at least one of the two nodes is a collaborating node, the collaborator can ignore the request and wait for the file requester to start the node selection process again, going back to the original situation.

Denote σ as the probability that the file requester is identified as the predecessor by the collaborator.

$$\begin{aligned} \sigma &= P(H_1) = P(I') = \frac{P(I)}{1 - P(C_{1,2})} \\ &= (\frac{c^2}{n^2})(\frac{n^2}{n^2 - 2c(n-c)}) = \frac{c^2}{n^2 - 2c(n-c)} \end{aligned} \tag{5}$$

Now since the collaborator will ignore the request if only one of them is chosen, H_{2+} can be given by (p is the forwarding probability)

$$\begin{aligned} P(H_{2+}) &= \sum_{i=1}^{\infty} \left(\frac{p(n-c)^2}{n^2}\right)^i \sigma \\ &= \left(\frac{p(n-c)^2}{n^2 - p(n-c)}\right)\sigma \\ &= \frac{p\sigma(n-c)^2}{(n^2 - p(n-c)^2)} \end{aligned} \tag{6}$$

The probability that a non-collaborating node will appear before a pair of collaborating nodes, σ', will be:

$$\sigma' = \frac{P(H_{2+})}{n-c} = \frac{p\sigma(n-c)}{(n^2 - p(n-c)^2)} \tag{7}$$

Let T be the total number of rounds that the file requester successfully selected two nodes (case (1): both are attackers, case (2): both are non-attackers). And $B(T, \sigma)$ be a binomial distribution with parameters T and σ.

We make use of a similar approach described in [26], in which we improve the estimate of the number of rounds needed in identifying the requester (reducing the number of rounds needed by 20% for the same level of confidence). The probability that the file requester and a non-requester will appear before the first collaborator in a round is given by $\sigma = \frac{c^2}{n^2 - 2c(n-c)}$ and $\sigma' = \frac{p\sigma(n-c)}{(n^2 - p(n-c)^2)}$ respectively. We need to make sure that the file requester will appear more than that of any non-requester.

To make sure that the file requester will appear at least $(1-\delta)T\sigma$ times, where $0 < \delta < 1$, we apply Chernoff Bound [18]:

$$Pr\left\{B(T, P(H_1)) > (1-\delta)T\sigma)\right\} \quad < \quad e^{-\frac{T\sigma\delta^2}{2}} \tag{8}$$

When $T = \frac{2}{\sigma\delta^2}\ln n$, with probability less than $\frac{1}{n}$, the file requester will appear less than $(1-\delta)T\sigma$ times.

Also we need to make sure that any non-requester will not appear more than $(1-\delta)T\sigma$ times, thus we set:

$$\begin{aligned}(1-\delta)T\sigma &= (1+\delta')T\sigma' \\ \delta' &= \frac{\sigma(1-\delta)}{\sigma'} - 1 \end{aligned} \quad (9)$$

We apply Chernoff Bound again to make sure the non-requester will not appear too frequently.

$$\begin{aligned}Pr\{B(T,\sigma') < (1+\delta')T\sigma'\} &< 2^{-\delta' T\sigma'} \\ &= 2^{-T(\sigma(1-\delta)-\sigma')} \end{aligned} \quad (10)$$

When $T = \frac{a}{\sigma(1-\delta)-\sigma'}\lg n$, where a is a constant, with probability $\frac{1}{n^a}$, a given non-requesting node will show up to the collaborator more than $(1+\delta')T\sigma'$ times. Now we have $(n-c)$ such non-requesting nodes in the network. By setting $a = \log_n(n-c)+1$ and applying Union Bound, there will be less than $\frac{1}{n}$ chance (since $\frac{n-c}{n^a} = \frac{1}{n}$) that any non-initiator nodes will show up more than $(1+\delta')T\sigma'$ times.

We have $T = \frac{2}{\sigma\delta^2}\ln n$ and $T = \frac{a}{\sigma(1-\delta)-\sigma'}\lg n$. Therefore,

$$\begin{aligned}\frac{2\ln n}{\sigma\delta^2} &= \frac{a\lg n}{\sigma(1-\delta)-\sigma'} \\ 2\ln n\sigma - (2\ln n\sigma)\delta - 2\ln n\sigma' &= (a\lg n\sigma)\delta^2 \\ (a\lg n\sigma)\delta^2 + (2\ln n\sigma)\delta + (2\ln n\sigma' - 2\ln n\sigma) &= 0 \end{aligned} \quad (11)$$

Solving the quadratic equation yields δ. Now with probability $\frac{1}{n}$, the initiator will appear fewer than $T = (1-\delta)\sigma$ times, and with probability $\frac{1}{n}$, there will be one or more non-requester appearing more than $T = (1+\delta')\sigma' = (1-\delta)\sigma$. Thus the probability that the file requester will be identified wrongly is $\frac{1}{n^2}$.

Suppose $n = 1000, c = 100, p = 0.5$, we will obtain $\delta = 0.5562$. If we use a similar analysis method for Crowds, we would need a total of 447 rounds in order to identify the initiator. The chance that a non-initiator will appear more frequently than the initiator will be about $\frac{1}{n^2}$. (In the analysis of Crowds, $\sigma = \frac{c}{n}$ and $\sigma' = \frac{cp}{n^2-np(n-c)}$). If we use the analysis for our proposed scheme, a total of 3660 rounds are needed to identify the file requester with the same failure probability. It may not be very realistic if we want such a low probability of failure (about 0.00001). We can further improve the method by replacing the failure probability with $\frac{1}{k^2}$, where k is any constant.

Similar to the steps given above, we obtain $T = \frac{2}{\delta^2}\ln k$, $T = \frac{a}{\sigma(1-\delta)-\sigma'}\lg k$ and the following quadratic equation

$$(a\lg k\sigma)\delta^2 + (2\ln k\sigma)\delta + (2\ln k\sigma' - 2\ln k\sigma) = 0 \quad (12)$$

The probability of failure is $\frac{1}{10^2}$ with $k = 10$. Using the same parameters $n = 1000, c = 100, p = 0.5$, in Crowds a total of 148 rounds, while in our scheme a total of 1220 rounds is needed. Figure 5 shows the number of rounds needed for Crowds and MIX-Crowds.

In figure 5, we only show the results for $p = 0.5$. In contrast, the number of rounds needed for different values of p in Crowds and MIX-Crowds are almost the same (the difference

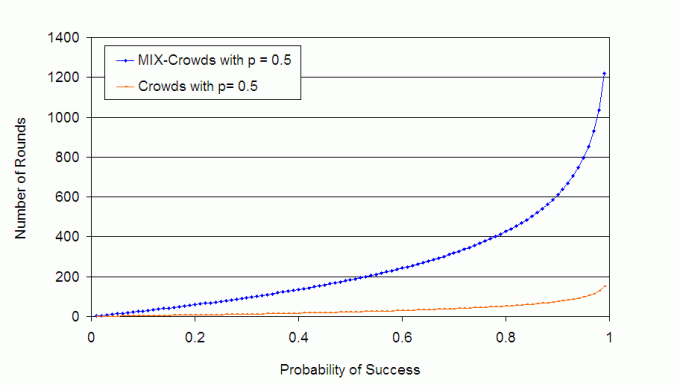

Fig. 5. The relationship between the probability of success and number of rounds for Crowds and MIX-Crowds with $p = 0.5$

in number of rounds is less than several rounds). The main reason is that it is quite unlikely that a non-requester node shows up more than a requesting node(as it can be seen that σ is much larger than σ').

We can explore further the benefit of increasing p using Reiter and Rubin's definition of anonymity [20]. A number of anonymity degrees are defined. Only three are useful in analyzing most peer-to-peer anonymous systems, in file retrieval systems context, namely

1) **Beyond Suspicion** - If an collaborator receives a message(request) from another node, the node will be no more likely to be the file requester (originator) of the message than any other nodes in the system.
2) **Probable Innocence** - If an collaborator receives a message(request) from another node, the node will be no more likely to be the file requester (originator) than not to be the file requester.
3) **Possible Innocence** - If an collaborator receives a message(request)from another node, the node will be more likely to be the file requester than not to be the file requester.

In our case, suppose N_R sends out a file request to another node. Suppose if N_1 and N_2 are both collaborators, and if $p = 0.5$, it can only be about 50% certain that N_R is the file requester. It may be possible that N_R is just forwarding a request for others. If 10% of the nodes are colllaborators, on average the predecessor will be identified every 82 rounds, but the number of rounds will increase dramatically if we want to make sure that the file requester will appear enough (two or more) times.

The number of rounds gives a rough estimation of how long it is needed to identify the file requester of a particular file. On the other hand, the degree of anonymity gives the collaborator an idea of how likely the predecessor is the file requester.

In Crowds, the content returned from the web server is only useful for the initiator. In our scheme, if $p = 0.5$, on average two other nodes will have a copy of the requested file after the transfer is done. As the file requester wants to stay anonymous, it will not share the file. In this case the two

nodes may further transfer the file to others by registering it in the overlay network (after some random delay to prevent the path from being revealed). However, some of the nodes may not share the file. If p is small, it is likely that the file will disappear in the network. At this stage, the file publisher N_P has to publish the file again using some anonymous publishing method, increasing its risk of being identified.

B. Extending the Scheme

In the above procedure, the content of the file together with the predecessor will only be revealed if all the chosen nodes (e.g., N_3 and N_4 in figure 4) are controlled by the collaborator. Even if the predecessor is revealed, the collaborator cannot be sure that it is the original requester (the assumption is that different files are being transfered in the system and delay between nodes is long enough so that it is difficult for the collaborator to carry out timing analysis).

Now if we increase the number of nodes chosen in each forwarding decision from 2 to m. Then,

$$P(C_{m,m}) = \left(\frac{c}{n}\right)^m = \frac{c^m}{n^m} \tag{13}$$

The chance that at least one of the m nodes is compromised is:

$$P(C_{1..m-1,m}) = 1 - \left(\frac{c}{n}\right)^m - \left(\frac{n-c}{n}\right)^m \tag{14}$$

Similar to the previous analysis above, we will obtain,

$$\sigma = P(H_1) = P(I') = P(I) + P(C_{1..m-1,m})P(I')$$

$$\sigma = \frac{P(I)}{1 - P(C_{1..m-1,m})} \tag{15}$$

$$\begin{aligned} P(H_{2+}) &= \sum_{i=1}^{\infty} \left(\frac{p(n-c)^m}{n^m}\right)^i \sigma \\ &= \frac{p\sigma(n-c)^m}{(n^m - p(n-c)^m)} \end{aligned} \tag{16}$$

$$\sigma' = \frac{P(H_{2+})}{n-c} = \frac{p\sigma(n-c)^{m-1}}{(n^m - p(n-c)^m)} \tag{17}$$

Substituting σ and σ' back to equation 12 to solve for δ, figure 6 shows the relationship between the number of rounds needed and the probability of success for different values of m.

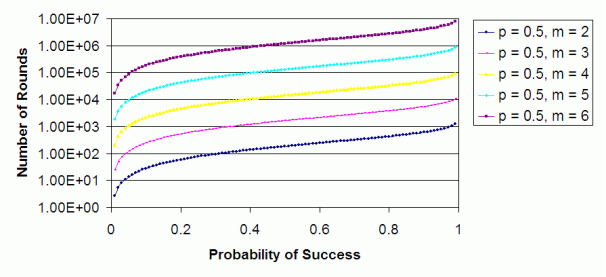

Fig. 6. The relationship between the probability of success and number of rounds for MIX-Crowds with $p = 0.5$ and different values of m

We can see that increasing m will make the success of predecessor attack much more difficult (increase in power order). However, the following should be noticed:

1) The intermediate nodes will not know the content of the file. This is good for anonymity. However, this also decreases the number of nodes registering the file on the network.
2) It is more likely that at least one or more collaborators will be chosen in each step. As a result, another group of nodes has to be chosen. This will increase the transfer time significantly if the number of nodes controlled by the collaborator is large.

C. Comparison with Other Anonymous Communication System

Table V summarize a general comparison between MIX-Crowds and some well-known anonymous communication systems. The probability of innocence indicates the approximate probability that when a node receives a message from another node, how certain it can deduce that the node is the initiator of the message (although the message content may not be revealed). For MIX systems, the second or later MIXes will know that the previous MIX is not the original sender of the message, thus it is 0 in the table. Probability of all nodes chosen by N_R is compromised is same as $P(H_1)$ in our analysis. Number of bits required to send out 1 bit data indicates the efficiency of the network. For example in MIX like systems (e.g. Onion Routing [10]), to retrieve the content of a web page with size L bytes, at least mL bytes would be required to be transferred on the network.

We can see that for schemes not using a directory server to store public key information (Crowds, Hordes, Freenet), the content transferred by the predecessor will always be revealed. On the other hand, MIX networks encrypt the message with layered encryption from the first node to the last node, thus all intermediate nodes will not know about the content of the transfer. The low efficiency of Freenet is due to the fact that searching is done in a partially unstructured manner. MIX-Crowds combines the idea of both Crowds and MIX. The reason why other anonymous communication systems like onion routing is used because no caching will be performed in the intermediate nodes. For contents that needed to be cached (e.g., files, frequently requested web documents), MIX-Crowds provides a balance between anonymity and confidentiality (among intermediate nodes) with the help of public key systems and the directory server. For Crowds, a blender is already needed to store the list of nodes, so the additional cost for storing the public key of the nodes will not be high. The performance of Crowds with $p = 0.8$ is similar to that in MIX-Crowds with $p = 0.6$ and $m = 2$. It is harder for the collaborator to identify the requester together with the content of the request in MIX-Crowds as compare with Crowds of similar level of efficiency.

In this section, we have analyzed the anonymity of MIX-Crowds based on [26] with some modifications and extension. We have also compared the anonymity of MIX-Crowds with

System Name	Probability of Innocence $(0-1]$	Probability of all nodes chosen by N_R are compromised	Number of bits required to sent out 1 bit data
Crowds [20]	$\frac{p(n-c-1)}{n}$	$\frac{c}{n}$	$O(\frac{1}{1-p})$
Hordes [17]	$\frac{p(n-c-1)}{n}$	$\frac{c}{n}$	O(No. of nodes in the multicast group)
Freenet [8]	depends on key closeness	$\approx \frac{c}{n}$	O(log n)
MIX [7]	0	$\approx (\frac{c}{n})^m$	$O(m)$
MIX-Crowds	$\frac{p(n-c-1)}{n}$	$\approx (\frac{c}{n})^m$	$O(\frac{m}{1-p})$

TABLE V
COMPARISON OF MIX-CROWDS AND OTHER ANONYMITY COMMUNICATION SCHEMES

other anonymous communication systems and carried out security analysis under some common types of attacks.

IV. CONCLUSION

A new anonymous communication network, MIX-Crowds, based on the idea of MIX and Crowds is described in this paper. The network can be used for different purposes, especially when the content is needed to be cached and kept confidential to other intermediate nodes.

In our proposed system, file searching and publishing can be done with the help of an overlay network. By using the idea of MIX, the file content is kept secret from the previous nodes, while the retrieved file can be cached in the network by using the idea of Crowds.

We modify the analysis method described in [26] to achieve a lower number of rounds needed for the predecessor attack with various levels of probability of success. Anonymity analysis of MIX-Crowds for the modified scheme is given in Section III. The result shows that it is harder for the collaborators to find out the requester of a particular file F based on the predecessor attack. Also by separating files into four classes according to their size, transfer time for most files can be shortened, while that for large files only increases slightly.

There are still rooms of improvement for MIX-Crowds. In particular, we need to address some of the issues like bottleneck on the directory server, the better use of the overlay network and how to increase the number of users in MIX-Crowds to provide better anonymity. For analysis, as MIX-Crowds is based on the concepts of MIX and Crowds, defense against timing analysis should also be considered.

REFERENCES

[1] *Anonymizer*. http://www. anonymizer. com/.
[2] *Napster*. http://www. napster. com/.
[3] *The CAPTCHA Project*. http://www. captcha. net/.
[4] *Yahoo!* http://www. yahoo. com.
[5] J. Camenisch. Design and implementation of the idemix anonymous credential system. *Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30, 2002.
[6] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.S. Wallach. Secure routing for structured peer-to-peer overlay networks. *ACM SIGOPS Operating Systems Review*, 36(si):299, 2002.
[7] D.L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
[8] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Workshop on Design Issues in Anonymity and Unobservability*, 320, 2000.
[9] R. Dingledine, M.J. Freedman, and D. Molnar. The Free Haven project: Distributed anonymous storage service. *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 67–95, 2000.
[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. *Proceedings of the 13th USENIX Security Symposium*, 2, 2004.
[11] John Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.
[12] C. Gulcu and G. Tsudik. Mixing E-mail with Babel. *Network and Distributed System Security, 1996., Proceedings of the Symposium on*, pages 2–16, 1996.
[13] K.P. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 314–329, 2003.
[14] M. Jakobsson. Flash mixing. *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pages 83–89, 1999.
[15] D. Kesdogan, J. Egner, and R. Buschkes. Stop-and-go MIXes providing probabilistic anonymity in an open system. *Information Hiding Workshop*, 1998.
[16] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the Kazaa network. *Internet Applications. WIAPP 2003. Proceedings. The Third IEEE Workshop on*, pages 112–120, 2003.
[17] B.N. Levine. Hordes: a multicast based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.
[18] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
[19] A. Nandan, G. Pau, and P. Salomoni. GhostShare-reliable and anonymous P2P video distribution. *Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004. IEEE*, pages 200–210.
[20] M.K. Reiter and A.D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
[21] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.
[22] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Performance Comparison of the AES Submissions. *The Second AES Conference*, pages 15–34, 1999.
[23] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proceedings of the 2001 SIGCOMM conference*, 31(4):149–160, 2001.
[24] W.H. Tang. An Anonymity Scheme for File Retrieval Systems. *MPhil thesis, The University of Hong Kong, 2008*.
[25] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. *Proceedings of Eurocrypt*, 2003, 2003.
[26] M. Wright, M. Adler, B.N. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. *Network and Distributed System Security Symposium*, 2002.
[27] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, pages 5–14, 2002.
[28] L. Zou, EW Zegura, and MH Ammar. The effect of peer selection and buffering strategies on the performance of peer-to-peer file sharing systems. *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE International Symposium on*, pages 63–70, 2002.