Detecting and Defending against Malicious Attacks in the iTrust Information Retrieval Network

Yung-Ting Chuang, Isaí Michel Lombera, P. M. Melliar-Smith, L. E. Moser
Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
{*ytchuang, imichel, pmms, moser*}*@ece.ucsb.edu*

***Abstract*—This paper presents novel statistical algorithms for detecting and defending against malicious attacks in the iTrust information retrieval network. The novel detection algorithm determines empirically the probabilities of the exact numbers of matches based on the number of responses that the requesting node receives. It calculates analytically the probabilities of the exact numbers of matches and the probabilities of one or more matches when some proportion of the nodes have been subverted or are non-operational. It compares the empirical and analytical probabilities to estimate the proportion of subverted or non-operational nodes. The novel defensive adaptation algorithm then increases the number of nodes to which the metadata and the requests are distributed to maintain the same probability of a match when some of the nodes are subverted or non-operational as when all of the nodes are operational. Experimental results substantiate the effectiveness of the statistical algorithms for detecting and defending against malicious attacks.**

***Keywords*-decentralized distributed information retrieval; detecting defending malicious attacks; iTrust**

I. Introduction

Modern society depends on retrieval of information over the Internet. For reasons of efficiency, Internet search and retrieval exploits centralized search engines, and assumes that they are unbiased. Unfortunately, it is easy to cause an Internet search engine to conceal or censor information. The experience of history, and of today, demonstrates that we cannot depend on centralized search engines to remain unbiased. The moment at which society is most dependent on the free flow of information across the Internet might also be the moment at which information is most likely to be censored or suppressed.

The iTrust system [2], [13] is a decentralized and distributed information retrieval system, that is designed to defend against censorship of information on the Internet. In iTrust, metadata describing documents, and requests (queries) containing metadata (keywords), are randomly distributed to multiple participating nodes. The nodes that receive the requests try to match the keywords in the requests with the metadata they hold. Crucial performance parameters of iTrust are the numbers of nodes to which the metadata and the requests must be distributed to achieve a high probability of achieving a match. In a network of n participating nodes, distribution of the metadata and the requests to $2\sqrt{n}$ nodes results in a probability of a match that exceeds $1 - e^{-4} \approx 0.9817$. The communication cost of iTrust is greater than that of a centralized search engine but, if users are concerned about censorship or suppresion of information, they should be willing to incur that extra cost.

The decentralized and distributed nature of iTrust makes it very robust against malicious attacks that aim to prevent information retrieval. One specific kind of malicious attack is to insert into the network a large number of nodes that behave normally except that they do not match requests and metadata on certain topics. The appropriate response to such an attack is to increase the number of nodes to which the metadata and the requests are distributed, to restore the probability of a match to the desired level. This paper presents novel statistical algorithms for detecting such malicious attacks and for adaptively defending against them.

II. Design of iTrust

The iTrust information retrieval system is completely distributed, and involves no centralized mechanisms and no centralized control. We refer to the nodes that participate in an iTrust network as the *participating nodes* or the *membership*. Multiple iTrust networks may exist at any point in time, and a node may participate in multiple iTrust networks at the same point in time.

In iTrust, some nodes, the *source nodes*, produce information, and make that information available to other participating nodes. The source nodes produce metadata that describes their information, and distribute that metadata to a subset of the participating nodes chosen at random, as shown in Figure 1. The metadata are distinct from the information they describe, and include a list of keywords and the URL of the source of the information.

Other nodes, the *requesting nodes*, request and retrieve information. Such nodes generate requests (queries) that refer to metadata for the desired information, and distribute their requests to a subset of the participating nodes chosen at random, as shown in Figure 2.

The participating nodes compare the metadata in the requests they receive with the metadata they hold. If such a node finds a match, which we call an *encounter*, the matching node returns the URL of the associated information to the requesting node. The requesting node then uses the URL

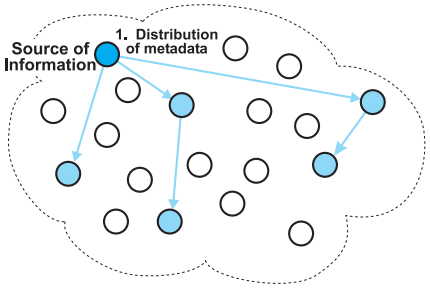

Figure 1. A source node distributes metadata, describing its information, to randomly selected nodes in the network.

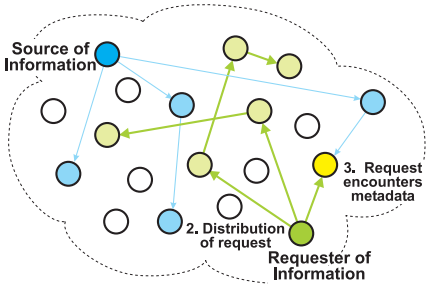

Figure 2. A requesting node distributes its request to randomly selected nodes in the network. One of the nodes has both the metadata and the request and, thus, an encounter occurs.

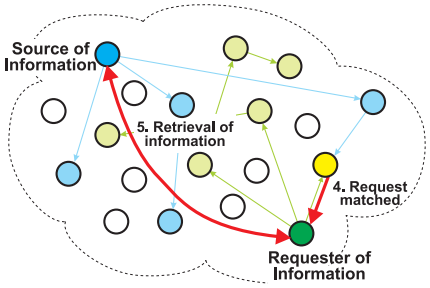

Figure 3. A node matches the metadata and the request and reports the match to the requester, which then retrieves the information from the source node.

to retrieve the information from the source node, as shown in Figure 3.

III. Implementation of iTrust

The iTrust system runs on laptop, desktop or server nodes over the Internet. There might be hundreds or thousands of iTrust nodes in a typical iTrust network. The primary goal of each iTrust node is to match a query it receives with metadata it holds and to respond with a URL for that resource, if an encounter occurs.

The iTrust implementation on a node consists of the Web server foundation, the application infrastructure, and the public interface. These three components interact with each other to distribute metadata and requests, and to retrieve resources from the nodes.

The iTrust implementation utilizes the Apache Web server along with several PHP modules and library extensions. In particular, it uses the standard session and logging modules, and the compiled-in cURL, SQLite, and HTTP PHP Extension Community Library (PECL) modules.

The iTrust system operates over HTTP and, thus, TCP/IP. As such, it establishes a direct connection between any two nodes that need to communicate; the iTrust implementation uses neither flooding nor random walks.

The iTrust system replicates both the metadata and the requests (queries). A source node randomly selects nodes for distribution of metadata, and a requesting node randomly selects nodes for distribution of requests. At each node, iTrust maintains a local index of metadata and URLs for the corresponding resources.

When a node receives a request, it compares the metadata keywords in the request (query) against an SQLite database consisting of metadata and URLs. If the metadata keywords in the request match locally stored metadata, the node sends a response containing the URL corresponding to the metadata to the requester. When the requesting node receives the response, it uses the URL in the request to retrieve the information from the source node.

IV. Foundations

A. Assumptions and Notation

We assume that all of the participating nodes in an iTrust network have the same membership set. Moreover, we assume that the underlying Internet delivers messages reliably and that the nodes have enough memory to store the source files and metadata that the nodes generate and receive. If a node receives a request and it holds metadata matching the request, we say that a *match* occurs.

The primary parameters determining the performance of the iTrust system are:

- n: The number of participating nodes (*i.e.*, the size of the membership set)
- x: The proportion of the n participating nodes that are operational (*i.e.*, $1 - x$ is the proportion of non-operational nodes)
- m: The number of participating nodes to which the metadata are distributed
- r: The number of participating nodes to which the requests are distributed
- k: The number of participating nodes that report matches to a requesting node.

B. Probabilistic Analysis

Our probabilistic analysis of iTrust is based on the hypergeometric distribution [6], which describes the number of successes in a sequence of random draws from a finite population *without* replacement. Thus, it differs from the binomial distribution, which describes the number of successes for random draws *with* replacement. In iTrust, the probability of exactly k matches follows a hypergeometric distribution with parameters n, x, m and r, and is given by:

$$P(k) = \frac{\binom{mx}{k}\binom{n-mx}{r-k}}{\binom{n}{r}} \tag{1}$$

for $mx + r \leq n$ and $k \leq \min\{mx, r\}$.

Thus, if the iTrust membership set contains n participating nodes, the proportion of the participating nodes that are operational is x, the metadata are delivered to m participating nodes, and a request is delivered to r participating nodes, then the probability of *exactly* k matches is given by:

$$P(k) = \frac{(\frac{mx}{k}\frac{mx-1}{k-1}\cdots\frac{mx-k+1}{1})(\frac{n-mx}{r-k}\frac{n-mx-1}{r-k-1}\cdots\frac{n-mx-r+k+1}{1})}{(\frac{n}{r}\frac{n-1}{r-1}\cdots\frac{n-r+1}{1})} \tag{2}$$

for $mx + r \leq n$ and $k \leq \min\{mx, r\}$.

If the iTrust membership set contains n participating nodes, the proportion of the participating nodes that are operational is x, the metadata are delivered to m participating nodes, and a request is delivered to r participating nodes, then the probability of *one or more* matches is given by:

$$P(k \geq 1) = 1 - \frac{\left(\frac{n-mx}{r}\,\frac{n-mx-1}{r-1}\cdots\frac{n-mx-r+1}{1}\right)}{\left(\frac{n}{r}\,\frac{n-1}{r-1}\cdots\frac{n-r+1}{1}\right)} \qquad (3)$$

for $mx + r \leq n$.

For an iTrust network with $n = 1000$ nodes and $x = 1.0$, 0.7, $0.4, 0.2$ operational nodes, Figure 4 shows the results for the probabilities of one or more matches, obtained from Equation (3), as the number of nodes to which the metadata and the requests are distributed increases.

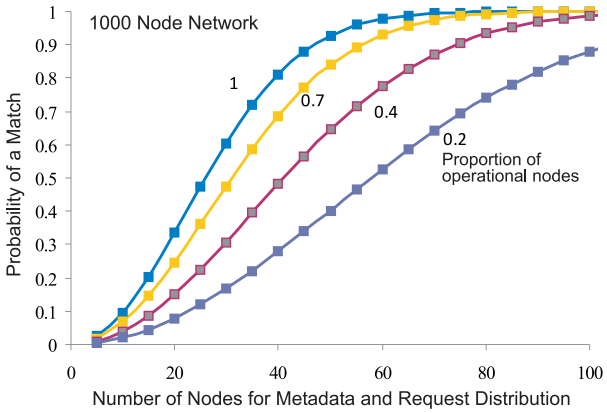

Figure 4. The probabilities of one or more matches as the number of nodes to which the metadata and the requests are distributed increases, for different proportions of operational nodes.

V. Detecting Malicious Attacks

For given values of n, m and r, the algorithm for detecting malicious attacks collects statistical data on the number of responses that a requesting node has received for a number of requests. Then, it computes the analytical probabilities of the exact number of matches for n, m and r and for different values of x. Finally, it compares the empirical probabilities against the analytical probabilities of exact number of matches to estimate the proportion x' of operational nodes in the iTrust network.

For example, consider an iTrust network with $n = 1000$ participating nodes where both the metadata and the requests are distributed to $m = r = 60$ nodes. Figure 5 shows the probabilities of the exact number of matches for $x = 1.0, 0.7, 0.4, 0.2$ operational nodes. If the detection algorithm observes a curve like the 0.2 shown in Figure 5, it detects that there is a large number of subverted or non-operational nodes and that the network is under attack.

In Figure 5, it is easy to distinguish the behavior of the iTrust network with many operational nodes from its behavior with few operational nodes. For example, the curves for $x = 0.2$ and $x = 1.0$ are quite different. Although the probability of exactly 1 match for $x = 0.2$ is greater than

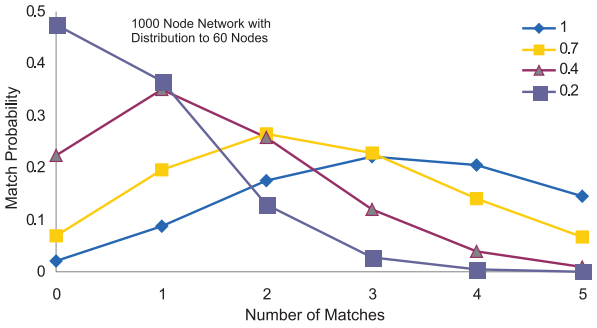

Figure 5. The probabilities of the exact number of matches for various proportions of operational nodes.

the corresponding probability for $x = 1.0$, the probabilities of exactly k matches, $k \geq 2$, for $x = 1.0$ are greater than the corresponding probabilities for $x = 0.2$.

The detection algorithm uses Pearson's chi-squared goodness-of-fit test [9] to determine which of the analytical curves best matches the empirical data for the probability of a match. The χ^2 statistic is given by:

$$\chi^2 = \sum_{k=1}^{K} \frac{(o_k - e_k)^2}{e_k} \qquad (4)$$

where K is the number of buckets into which the observations fall, o_k is the number of observations that fall into the kth bucket, and e_k is the expected (theoretical) number of observations for the kth bucket obtained from Equation (2).

Using the observations, the detection algorithm evaluates χ^2 for $x = 1.0, 0.7, 0.4, 0.2$, compares those values of χ^2, and then chooses the smallest of them. The value of x that corresponds to the smallest value of χ^2 is the algorithm's best estimate of the proportion x' of operational nodes.

VI. Defending against Malicious Attacks

The algorithm for defending against malicious attacks increases the number m of nodes to which the metadata are distributed and the number r of nodes to which the requests are distributed. It increases m and r to achieve the same probability of a match for $x < 1.0$ as for $x = 1.0$.

As an example, consider the curves for $x = 1.0$ and $x' = 0.7$ shown in Figure 6 for $n = 1000$ nodes. For $x = 1.0$ and a specific value of $m = r$, say $m = r = 60$, the algorithm first computes the probability of one or more matches to obtain $y_0 = P(k \geq 1)$. That is, it determines the value of y_0 for the point on the $x = 1.0$ curve corresponding to $m = r = 60$. Thus, for $n = 1000$, $x = 1.0$ and $m = r = 60$, it calculates $y_0 = 0.978298 = P(k \geq 1)$ from Equation (3).

If the defensive adaptation algorithm determines from empirical evidence that $x' = 0.7$, then from the calculated value of y_0 it determines the value of $m' = r'$ corresponding to y_0 on the $x = 0.7$ curve. That is, it iteratively solves the equation $y_0 = P'(k \geq 1)$ with $n = 1000$, $x' = 0.7$ and $m' = r'$ in Equation (3). For this example, the defensive adaptation algorithm finds that $m' = r' = 72$, using the iterative algorithm shown in Figure 7.

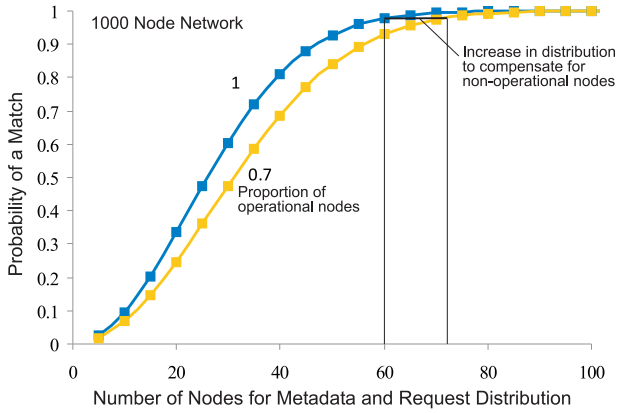

Figure 6. Increase in the distribution of metadata and requests to compensate for non-operational nodes in order to achieve the same probability of a match.

Initialize n, x' $m' = m$ $r' = r$, y_0
Repeat
$m' = m' + 1$; $r' = r' + 1$

$$y' = 1 - \frac{(n - m'x')}{(n)} \frac{(n - m'x' - 1)}{(n-1)} \cdots \frac{(n - m'x' - r' + 1)}{(n - r' + 1)}$$

until $y' > y_0$
return m', r'

Figure 7. Algorithm for iteratively finding the values of m' and r' to maintain the probability of a match if some of the nodes are subverted.

VII. Experimental Evaluation

To evaluate the ability of the detection algorithm to estimate the value of x, we performed experiments using a simulation of iTrust. The simulation allows us to evaluate the accuracy of the detection algorithm, whereas a real-world deployment of iTrust would not allow us to do so, because in the simulation we can control the value of x. We use the detection algorithm to obtain an estimate x' of the proportion of operational nodes and determine the number of times the estimate is correct by comparing x' with x.

A. Accuracy

To determine the accuracy of the detection algorithm, we perform the following experiments for each value of $x = 1.0, 0.7, 0.4, 0.2$. We consider a window of w requests. For each request in the window, the requester sends r request messages to randomly selected nodes in the iTrust network. If a node holds metadata that match the metadata in a request message it receives, it sends a response to the requester. We cannot use request messages that return $k = 0$ responses, because $k = 0$ responses can arise not only when the metadata and request messages are distributed to disjoint subsets of nodes, but also when there is no metadata that matches the metadata in a request. Consequently, we exclude the request messages that return $k = 0$ responses. Moreover, for $6 \leq k \leq r$, the probabilities are neglibly small, so we exclude those request messages too.

For each request in the window, the algorithm determines the number of responses the requester received for that request. To determine the observed probability $P'(k)$ for the window, $1 \leq k \leq r$, the algorithm determines the number $N(k)$ of requests in the window for which it received k responses and then divides that number by w, *i.e.*,

$$P'(k) = \frac{N(k)}{w} \tag{5}$$

and renormalizes the $P'(k)$, by dividing by $\sum_{k=1}^{5} P'(k)$.

From the analytical formula (2), the algorithm calculates $P(k)$ using the values of n, m, r and x and then renormalizes the $P(k)$, by dividing by $\sum_{k=1}^{5} P(k)$. It then applies the chi-squared formula (4) to the renormalized $P'(k)$ and the renormalized $P(k)$ for $k = 1, 2, 3, 4, 5$ and chooses the particular value of x' ($x' = 1.0, 0.7, 0.4, 0.2$) for which the chi-squared value is the smallest.

In our experiments, we consider windows of sizes $w = 10, 20, \ldots, 100$. For each value of w, we repeat the experiment $S = 10,000$ times, and determine the number C out of S times for which the algorithm's estimate x' is correct. The accuracy $A(w)$ is given by:

$$A(w) = \frac{C}{S} \tag{6}$$

Figure 8 shows the accuracy $A(w)$ as a function of the window size $w = 10, 20, \ldots, 100$ for $x = 1.0, 0.7, 0.4, 0.2$ operational nodes in an iTrust network with $n = 1000$ nodes and $m = r = 60$. The figure shows that, for the larger window sizes, the accuracy for $x = 0.2$ is somewhat worse than that for the other values of x, which we are still investigating.

B. Response Time

Using the accuracy results obtained in the previous experiments, we then determine the mean response time of the detection algorithm. The mean response time $R(w)$, represented as the number w of requests in a window is given by:

$$R(w) = \sum_{i=1}^{T} iwA(w)(1 - A(w))^{i-1} \tag{7}$$

In Equation (7), the probability that the value of x is estimated correctly after the first window is $A(w)$, contributing $1wA(w)$ requests to the summation. The probability that the value of x is estimated incorrectly after the first window, and correctly after the second window is $A(w)(1 - A(w))$, contributing $2wA(w)(1 - A(w))$ requests to the summation, *etc.* Here, T is some large integer such that $TwA(w)(1-A(w))^{T-1}$ makes a negligible contribution to the sum.

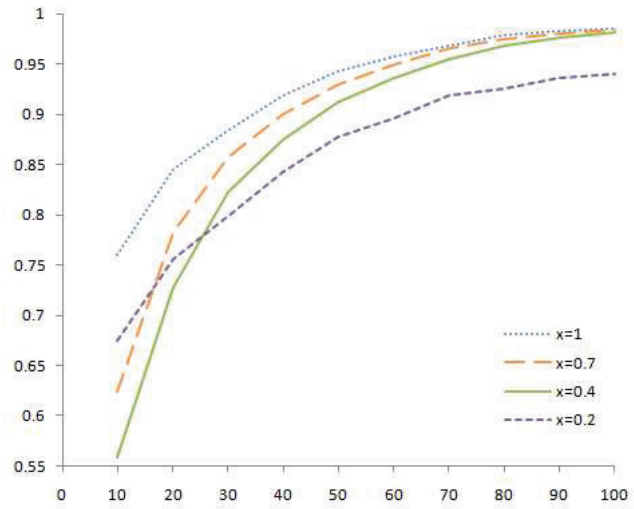

Figure 8. The accuracy of the detection algorithm for various window sizes and various proportions of operational nodes.

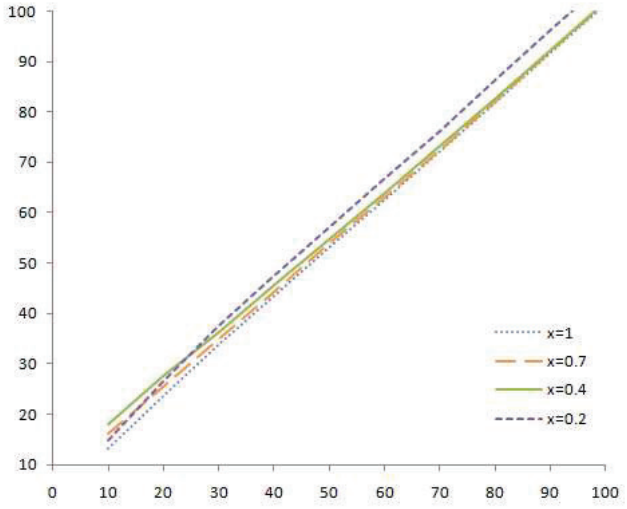

Figure 9. The mean response time of the detection algorithm for various window sizes and various proportions of operational nodes.

Simplifying Equation (7), we obtain:

$$\begin{aligned} R(w) &= wA(w)\sum_{i=1}^{T} i(1-A(w))^{i-1} \\ &= wA(w)\frac{1-(T+1)(1-A(w))^T+T(1-A(w))^{T+1}}{A(w)^2} \\ &= w\frac{1-(1+TA(w))(1-A(w))^T}{A(w)} \\ &\approx \frac{w}{A(w)} \end{aligned} \tag{8}$$

Figure 9 shows the mean response time $R(w)$ as a function of the window size $w = 10, 20, \ldots, 100$ for $x = 1.0, 0.7, 0.4, 0.2$ operational nodes in an iTrust network with $n = 1000$ nodes and $m = r = 60$. As the figure shows, the mean response time increases linearly with the window size w for each value of x. Moreover, the figure shows that, for different values of x and the same value of w, the mean response times are very close.

Comparing Figures 8 and 9 for specific values of w, we see that there is a tradeoff between the accuracy and the response time. A small value of w results in a low response time but also low accuracy. A large value of w increases the accuracy but also increases the response time. A compromise value of w, say $w = 40$, might be appropriate.

The response time, in seconds, depends not only on the number of requests in the window but also on the time interval between requests, *i.e.*, the time between issuing a request and obtaining the responses for that request plus the think time. If it takes s seconds between each request in the window, then the response time between requests is $sR(w)$ seconds. More frequent requests yield a lower response time. However, with a large number of participating nodes, frequent requests might overload the iTrust network.

VIII. Related Work

Peer-to-peer networks have been characterized as structured or unstructured [14]. The structured approach requires the nodes to be organized in an overlay network, based on a distributed hash table, tree, ring, *etc.* The unstructured approach uses randomization and/or gossiping to distribute the metadata (or data) and the requests to nodes in the network. The iTrust system uses the unstructured approach.

Gnutella [8], one of the first unstructured peer-to-peer networks, uses flooding of requests. A node makes a copy of a file when it receives the file it requested. If the query rate is high, nodes quickly become overloaded and the system ceases to function satisfactorily.

To address this problem, Ferreira *et al.* [7] use random walks, instead of flooding, to replicate objects and propagate queries. Gia [1] employs biased random walks to direct queries towards high-capacity nodes. iTrust does not use flooding or random walks, but rather distributes the metadata and the requests to nodes chosen uniformly at random.

Freenet [3] replicates an object at a node, even though the node did not request it. Nodes that successfully respond to requests receive more metadata and more requests, making it easy for a group of untrustworthy nodes to gather most of the searches into their group. The Adaptive Probabilistic Search (APS) method [18] likewise uses feedback from previous searches to direct future searches, instead of distributing the requests uniformly at random, like iTrust does.

Cohen and Shenker [4] show that square root replication is theoretically optimal for minimizing the search traffic. They replicate objects based on the access frequencies (popularities) of the objects. Lv *et al.* [12] also use square root replication, and adaptively adjust the replication degree based on the query rate. iTrust likewise exploits square root replication but distributes the metadata uniformly at random, so that popular nodes are not more vulnerable to attack.

BubbleStorm [17] replicates both queries and data, hybridizes random walks and flooding, and considers churn, leaves and crashes. Leng *et al.* [11] present mechanisms for maintaining the desired degree of replication in Bub-

bleStorm, when each object has a maintainer node. In iTrust, we use different techniques to maintain the desired degree of replication of the metadata and the requests.

Morselli *et al.* [15] describe an adaptive replication protocol with a feedback mechanism that adjusts the number of replicas according to the mean search length to determine if an object is replicated sufficiently. Their adaptive replication protocol is quite different from our defensive adaptation algorithm for iTrust.

Sarshar *et al.* [16] use random walks and bond percolation in power law networks with high-degree nodes. The high-degree nodes in such networks are subject to overloading and are vulnerable to targeted malicious attacks. They note: "protocols for identifying or compensating for such attacks, or even recovering after such an attack has disrupted the network are yet to be designed." We present such protocols in this paper.

Jesi *et al.* [10] identify malicious nodes in a random overlay network based on gossiping and put them on a blacklist. They focus on hub attacks in which colluding malicious nodes partition the network by spreading false rumors. iTrust does not use gossiping but, rather, uses random distribution of the metadata and the requests and, thus, is less subject to hub attacks.

Condie *et al.* [5] present a protocol for finding adaptive peer-to-peer topologies that protect against malicious peers that upload corrupt, inauthentic or misnamed content. Peers improve the trustworthiness of the network by forming connections, based on local trust scores defined by past interactions. Effectively, their protocol disconnects malicious peers and moves them to the edge of the network.

IX. Conclusions and Future Work

We have presented novel statistical algorithms for detecting and defending against malicious attacks in the iTrust information retrieval network. The detection algorithm determines empirically the probabilities of the exact numbers of matches based on the number of responses that a requesting node receives. Then, it compares the empirical probabilities with the analytical probabilities for the exact numbers of matches, to estimate the proportion of nodes that are subverted or non-operational. The defensive adaptation algorithm determines the increased number of nodes to which the metadata and the requests must be distributed to maintain the same probability of a match when some proportion of the nodes are subverted or non-operational, as when all of the nodes are operational. In future work, we will investigate other kinds of malicious attacks on iTrust, and detection and defensive adaptation algorithms for those kinds of malicious attacks.

Acknowledgment

This research was supported in part by U.S. National Science Foundation grant number NSF CNS 10-16193.

References

[1] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham and S. Shenker, "Making Gnutella-like P2P systems scalable," *Proc. 2003 ACM SIGCOMM Conf.*, Aug. 2003, pp. 407-418.

[2] Y. T. Chuang, I. Michel Lombera, L. E. Moser and P. M. Melliar-Smith, "Trustworthy distributed search and retrieval over the Internet," *Proc. 2011 Intl. Conf. Internet Computing*, July 2011, pp. 169–175.

[3] I. Clarke, O. Sandberg, B. Wiley and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Proc. Workshop Design Issues in Anonymity and Unobservability*, July 2000, pp. 46–66.

[4] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," *Proc. 2002 ACM SIGCOMM Conf.*, Aug. 2002, pp. 177–190.

[5] T. Condie, S. D. Kamvar and H. Garcia-Molina, "Adaptive peer-to-peer topologies," *Proc. 4th IEEE Intl. Conf. Peer-to-Peer Computing*, Aug. 2004, pp. 53–62.

[6] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 1, John Wiley & Sons, New York, NY, 1968.

[7] R. A. Ferreira, M. K. Ramanathan, A. Awan, A. Grama and S. Jagannathan, "Search with probabilistic guarantees in unstructured peer-to-peer networks," *Proc. 5th IEEE Intl. Conf. Peer-to-Peer Computing*, Aug. 2005, pp. 165–172.

[8] Gnutella, http://www.gnutella.wego.com

[9] P. E. Greenwood and M. S. Nikulin, *A Guide to Chi-Squared Testing*, Wiley Series in Probability and Statistics, Wiley-Interscience, Mar. 1996.

[10] G. P. Jesi, D. Hales and M. van Steen, "Identifying malicious peers before it's too late: A decentralized secure peer sampling service," *Proc. 1st Intl. Conf. Self-Adaptive and Self-Organizing Systems*, July 2007, pp. 237–246.

[11] C. Leng, W. W. Terpstra, B. Kemme, W. Stannat and A. P. Buchmann, "Maintaining replicas in unstructured P2P systems," *Proc. ACM CoNEXT 2008 Conf.*, Dec. 2008, pp. 19:1–19:12.

[12] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker, "Search and replication in unstructured peer-to-peer networks," *Proc. Intl. Conf. Supercomputing*, June 2002, pp. 84–95.

[13] I. Michel Lombera, Y. T. Chuang, P. M. Melliar-Smith and L. E. Moser, "Trustworthy distribution and retrieval of information over HTTP and the Internet," *Proc. Intl. Conf. Evolving Internet*, June 2011, pp. 7–13.

[14] J. Mischke and B. Stiller, "A methodology for the design of distributed search in P2P middleware," *IEEE Network*, vol. 18, no. 1, Jan. 2004, pp. 30–37.

[15] R. Morselli, B. Bhattacharjee, A. Srinivasan and M. A. Marsh, "Efficient lookup on unstructured topologies," *Proc. Conf. Principles of Distributed Computing*, July 2005, pp. 77–86

[16] N. Sarshar, P. O. Boykin and V. P. Poychowdhury, "Percolation search in power law networks: Making unstructured peer-to-peer networks scalable," *Proc. 4th IEEE Intl. Conf. Peer-to-Peer Computing*, Aug. 2004, pp. 2–9.

[17] W. W. Terpstra, J. Kangasharju, C. Leng and A. P. Buchman, "BubbleStorm: Resilient, probabilistic, and exhaustive peer-to-peer search," *Proc. 2007 ACM SIGCOMM Conf.*, Aug. 2007, pp. 49–60.

[18] D. Tsoumakos and N. Roussopoulos, "Adaptive probabilistic search in peer-to-peer networks," *Proc. 2nd Intl. Workshop Peer-to-Peer Systems*, Sept. 2003, pp. 102–109.